2nd Implementation

T3
김상재
박성준
이종빈
장서연

2nd Implementation

(인)

T3
김상재
박성준
이종빈
장서연

목차

MBTI 소개

1. MBTI 란?

- **M**icro **B**enchmark **T**est **I**nterface의 약자.
- 사용되는 시스템 콜은 Signal, Mutual Exclusion, Semaphore, IPC가 있다.
- POSIX 라이브러리를 포함한 커널 내부의 메커니즘 오버헤드를 분석하기 위하여 개발되었다.
- 최대한 시스템 콜 & POSIX 라이브러리의 오버헤드를 측정하도록 노력하였다.

2. MBTI의 기능

a. 코어(Core) 수에 따른 시스템 콜의 성능 측정
b. 프로세스(Process) 혹은 쓰레드(Thread) 수에 따른 시스템 콜의 성능 측정
c. 토폴로지(Topology)에 따른 시스템 콜의 성능 측정
d. 패턴(Pattern)의 반복 수에 따른 시스템 콜의 성능 측정
e. 성능 측정 결과값을 CSV 파일 형식으로 저장 및 PNG 파일 형식으로 그래프 산출

개발 취지

- 유저 레벨에서 호스트 소프트웨어의 성능 향상 요소를 판단하는 것이 아닌 시스템 레벨에서의 개선사항을 찾기 위한 것이다.

- 시스템 호출과 유저레벨간의 인터페이스를 구현하는 유저, 혹은 기존의 시스템 유저 레벨 최적화를 넘어 시스템 최적화를 꾀하는 유저에게 범용적으로 쓰일 수 있다.

- 시스템 호출의 다양한 환경에 따른 성능을 측정한다.
  - 코어의 개수
  - 프로세스 개수
  - 토폴로지 종류
  - 패턴의 반복횟수

개발 취지

MBTI는 실제 소프트웨어를 테스트하는 것이 아닌 사용자가 원하는 시스템 콜과 미리 정해진 시스템 콜들의 실행 동작 패턴을 설정하여 테스트 한다.

- 편의성

a. 테스트 환경에 대한 오버헤드를 줄일 수 있다.
b. 여러 환경에서 시스템 콜 성능 측정을 간단히 구성할 수 있다
c. CSV 파일 형식의 데이터로부터 성능 측정의 산출물을 도출해 낼 수 있다.

- 다양성

a. 사용자 필요에 따라 다른 환경에서 테스트의 성능 측정 산출물을 도출해 낼 수 있다.
b. 특정 시스템 콜에 집중해서 분석, 비교할 수 있다.

개발 목표

각 시스템콜 Micro Benchmark Suite는 공통적으로 다음과 같은 기능들을 가지고 있어야 한다.

a. 코어 수에 따른 시스템 콜의 성능을 측정한다.
b. 프로세스 혹은 스레드 수에 따른 시스템 콜의 성능을 측정한다.
c. 토폴로지 종류에 따른 시스템 콜의 성능을 측정한다.
d. 패턴의 반복 수에 따른 시스템 콜의 성능을 측정한다.
e. 성능 측정 결과값을 CSV 파일 형식으로 저장 및 그래프를 산출한다.

개발 환경

- CPU: AMD Ryzen ThreadRipper 2950x, Cores/Threads: 16-Cores, 16-Threads
- OS: Ubuntu 20.04 LTS x64
- Kernel Version: 5.4.0
- GCC: gcc (Ubuntu 9.3.0-10ubuntu2) 9.3.0
- Python3: Python Interpreter 3.8.2

역할 소개

- 김상재
  - Python Script 개발
  - Signal 측정 파트 개발
- 박성준
  - Message Queue (IPC) 측정 개발
- 이종빈
  - Pthread Mutex (Lock) 측정 개발
- 장서연
  - Semaphore (Lock) 측정 개발

구현 – Python Script

```
Select a testing type

1. Signal
2. IPC
3. Lock
0. exit
```

```
* Selected Mode: Signal
Select topology

1: Ping-pong
0. exit

Topology:
```

```
Number of cores (Max cores: 16): 16
Number of processes or processes' pairs: 256
Number of pattern iterations: 10000
Number of tests: 10
Variations of graphs (1: Per cores, 2: Per processes): 1

* Test Attribute

0. Number of cores: 16
1. Number of processes or processes' pairs: 256
2. Number of pattern iterations: 10000
3. Number of tests: 10
4. Variations of graphs: 1
5. Gaps: 1

Confirm? (Y/N/0: exit):
```

1. 시스템 콜 선택

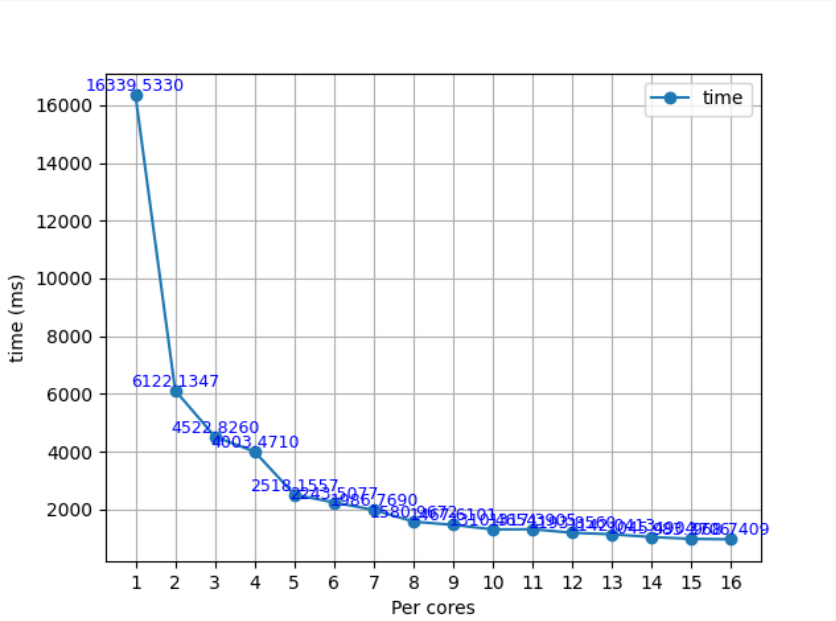
2. 토폴로지 선택

3. 테스트 속성 입력

구현 – Python Script

```
* CSV File is saved (file name: 20201109210122_SIG_PPT_COR.csv)
Save as png file? (Y/N):
```

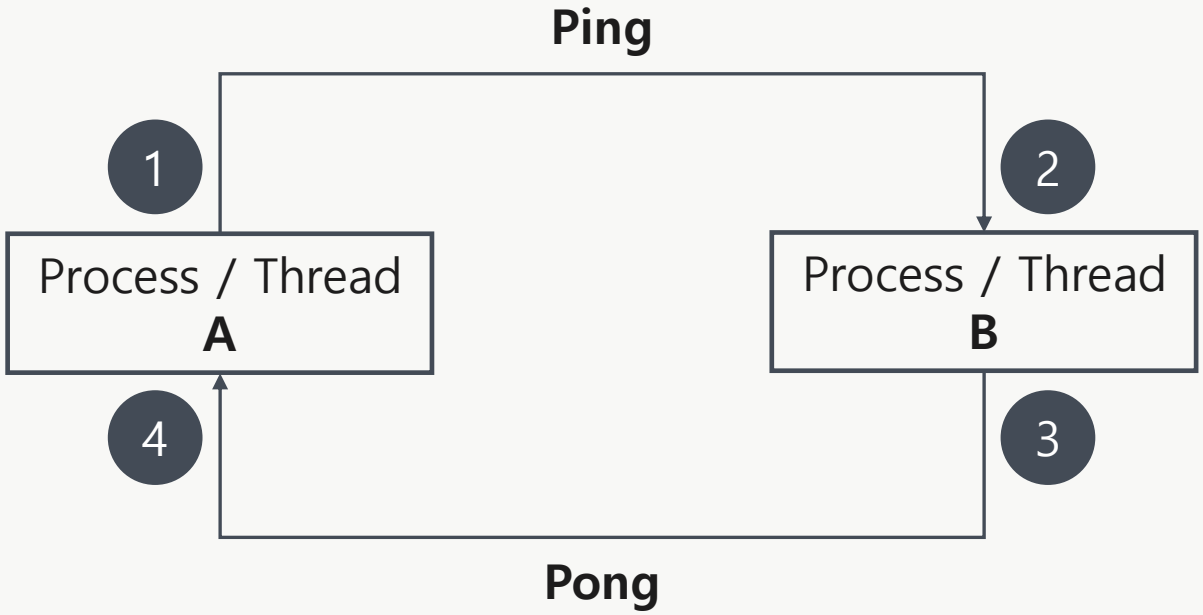
```
❯ cat 20201109210122_SIG_PPT_COR.csv
index,time
1,16339.532987400005
2,6122.134675700001
3,4522.8259598
4,4003.4710247000003
5,2518.1556687999996
6,2243.5076842999997
7,1986.7690234000002
8,1580.9671844
9,1467.6100735
10,1310.4653634000001
11,1317.3904529
12,1193.8560019
13,1142.0412828
14,1045.4903706999999
15,983.3706022
16,968.7409034
```



4. 측정 결과 CSV 파일 저장

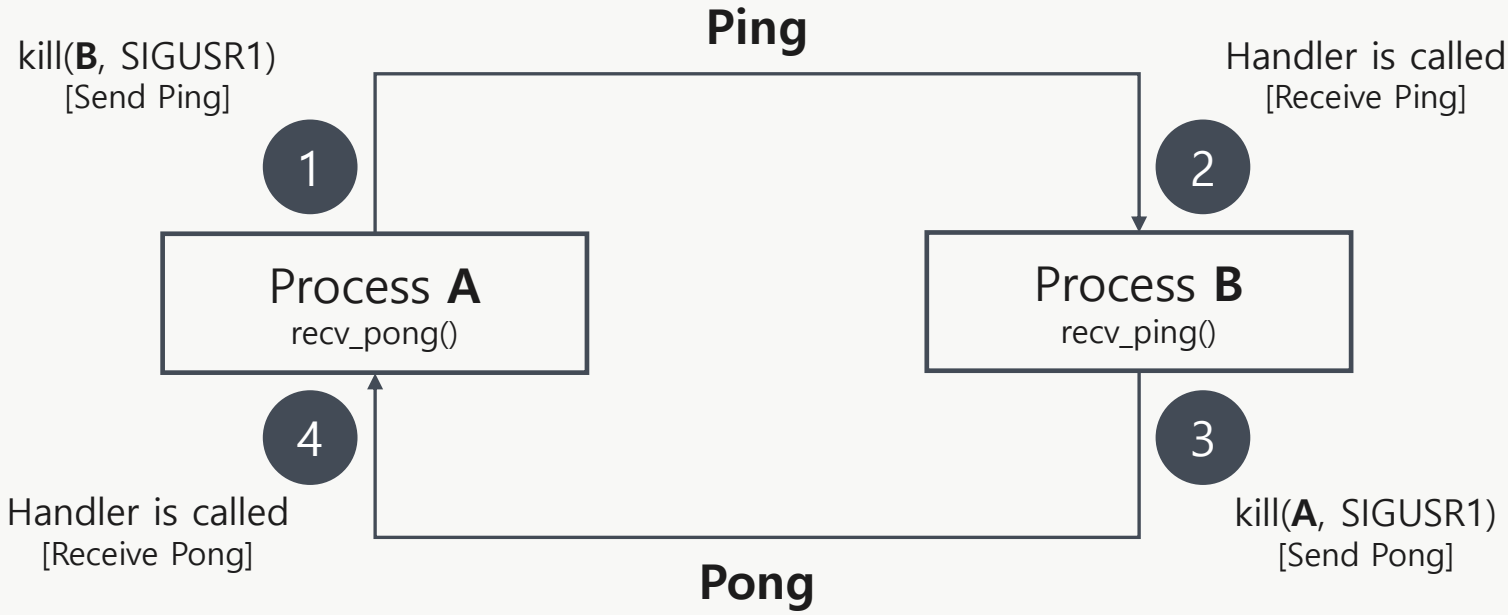
5. 측정 결과 PNG 파일 저장

구현 – Topology (Ping-pong) Overview



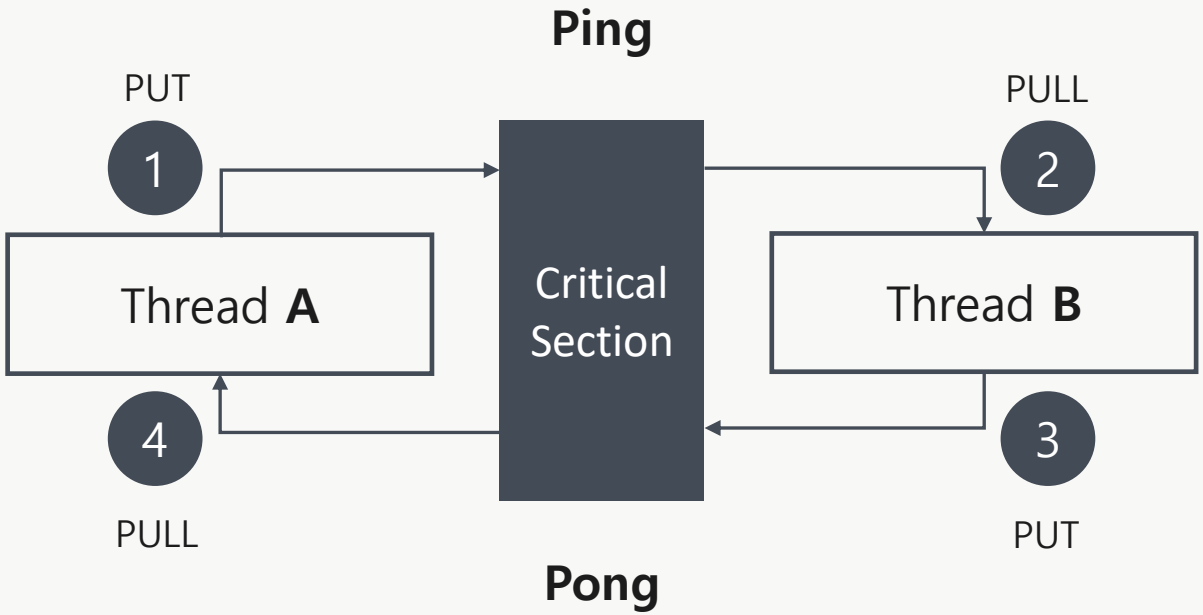
Ping	Pong
1. A에서 B로 Ping을 Send	3. B에서 A로 Pong을 Send
2. B에서 Ping을 Receive	4. A에서 Pong을 Receive

구현 – Signal (Ping-pong)



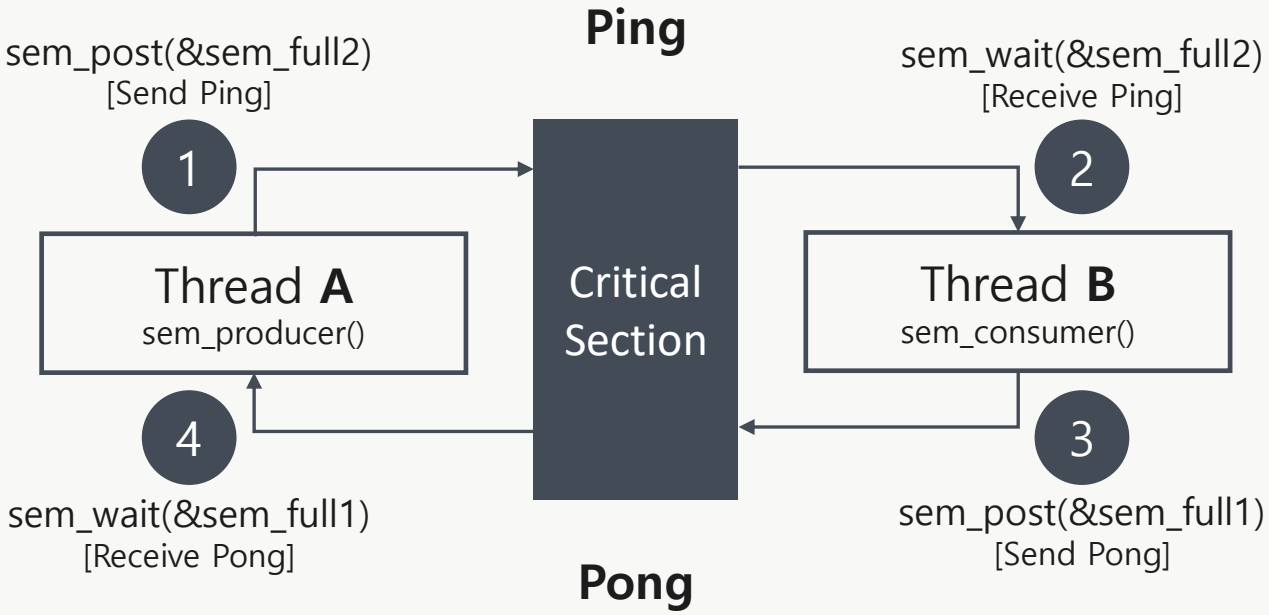
Ping	Pong
[Send Ping] 1. Process **A**에서 Process **B**로 kill 함수를 이용하여 SIGUSR1 Signal을 Send	[Send Pong] 3. Process **B**에서 Process **A**로 kill 함수를 이용하여 SIGUSR1 Signal을 Send
[Receive Ping] 2. Process **B**에서 SIGUSR1에 대한 Handler가 호출됨	[Receive Pong] 4. Process **A**에서 SIGUSR1에 대한 Handler가 호출됨

구현 – Mutex (Ping-pong)



Ping	Pong
[Send Ping] 1. Thread **A**가 Critical Section의 공유자료값에 값을 수정함. (값을 Put 함.)	[Send Pong] 3. Thread **B**가 Critical Section의 공유자료값에 값을 수정함. (값을 Put 함.)
[Receive Ping] 2. Wait 하던 Thread **B**가 깨어나서 Critical Section의 공유자료값을 Pull함.	[Receive Pong] 4. Wait 하던 Thread **A**가 깨어나서 Critical Section의 공유자료값을 Pull함.

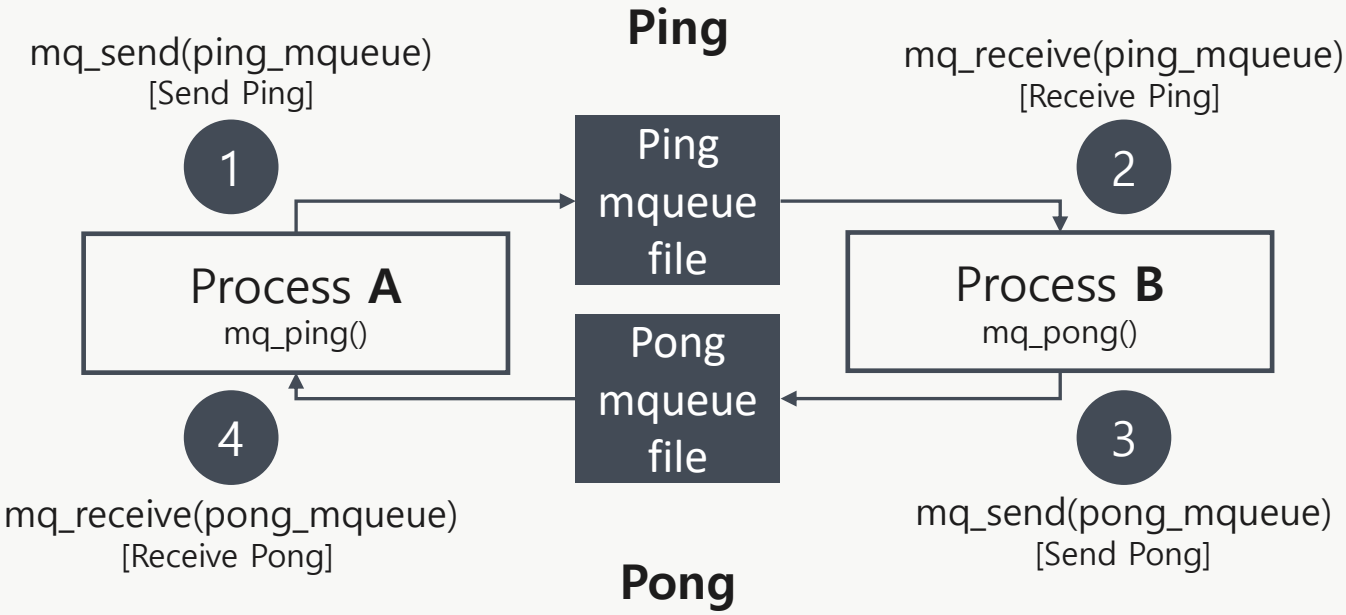
구현 – Semaphore (Ping-pong)



Ping	Pong
[Send Ping] 1. Thread **A**가 Critical Section의 공유자료값에 값을 수정함.	[Send Pong] 3. Thread **B**가 Critical Section의 공유자료값에 값을 수정함.
[Receive Ping] 2. Wait 하던 Thread **B**가 깨어나서 Critical Section에 접근.	[Receive Pong] 4. Wait 하던 Thread **A**가 깨어나서 Critical Section에 접근.

* sem_post()와 sem_wait() 사이에 공유 버퍼에 아이템 값을 넣고 소비하는 과정이 있습니다.

구현 – IPC (Ping-pong)

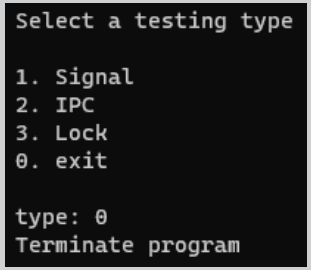
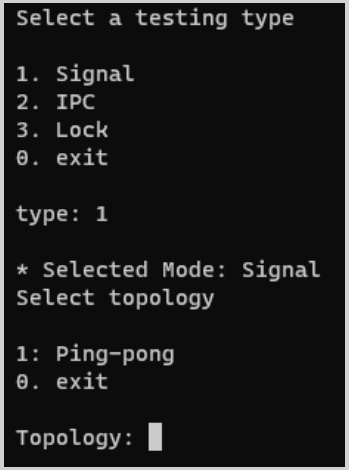
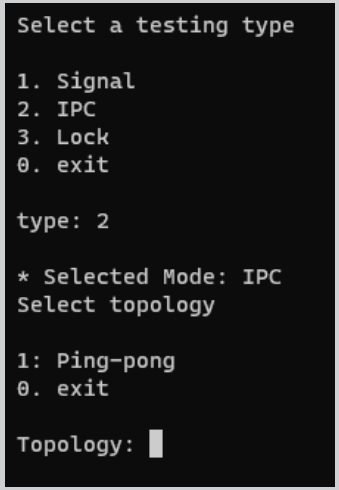
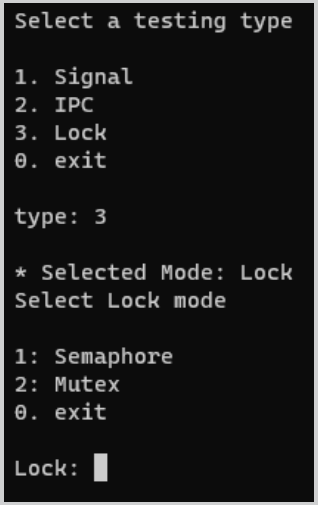


Ping	Pong
[Send Ping] 1. Process **A**에서 ping mqueue file에 mq_send()로 1Byte Data (Ping) Send	[Send Pong] 1. Process **B**에서 pong mqueue file에 mq_send()로 1Byte Data (Pong) Send
[Receive Ping] 2. Process **B**에서 ping mqueue file의 1Byte Data (Ping) 를 mq_receive()로 Receive	[Receive Pong] 2. Process **A**에서 pong mqueue file의 1Byte Data (Pong) 를 mq_receive()로 Receive

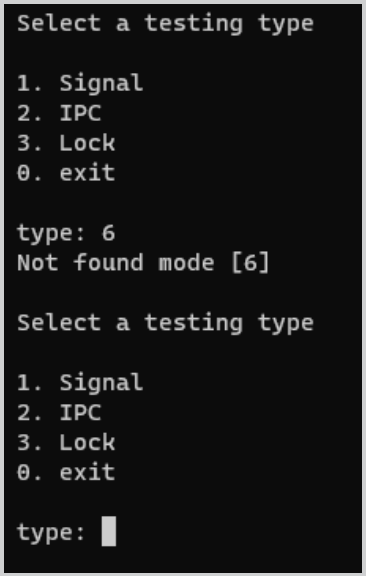
System Test - Python

Case No.	Req No.	Use case	Test description	Test Cases
1	3.2.1.1.	Select Mode	선택된 모드의 입력창으로 변경이 되는지	0 ~ 3 범위 내의 정수값
2			선택지에 존재하지 않는 값을 입력 시 오류메시지를 출력하는지	0 ~ 3 범위 외의 정수값
3	3.2.1.2.	Select Topology	선택된 토폴로지의 입력창으로 변경이 되는지	설정되 토폴로지 범위 내의 정수값
4			선택지에 존재하지 않는 값을 입력 시 오류메시지를 출력하는지	설정된 토폴로지 범위 외의 정수값
5	3.2.1.3.	Input Value	입력한 프로세스 개수 값, 테스트 반복 횟수 값, 코어 개수 값이 변수에 올바르게 입력 되는지	범위 내의 정수값
6			물리 코어 수를 초과한 경우 오류 메시지를 출력하는지	범위 외의 정수값
7			입력한 프로세스 값이 시스템이 임의로 설정한 값을 초과하면 오류 메시지를 출력한다.	시스템 설정 범위 외의 정수 값
8			입력한 테스트 반복 횟수 값이 범위를 벗어난 경우 오류 메시지를 출력하는지	시스템 설정 범위 외의 정수 값

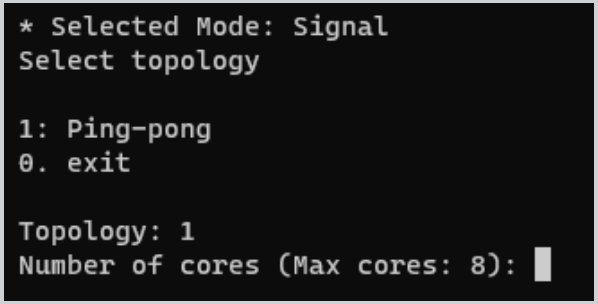
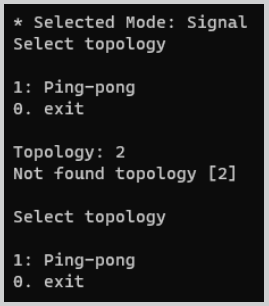
System Test – Python (Result)

Case No.	Test Result
1	   
	입력한 Type에 대한 선택지로 정상적으로 진행되는 것을 확인하였습니다.

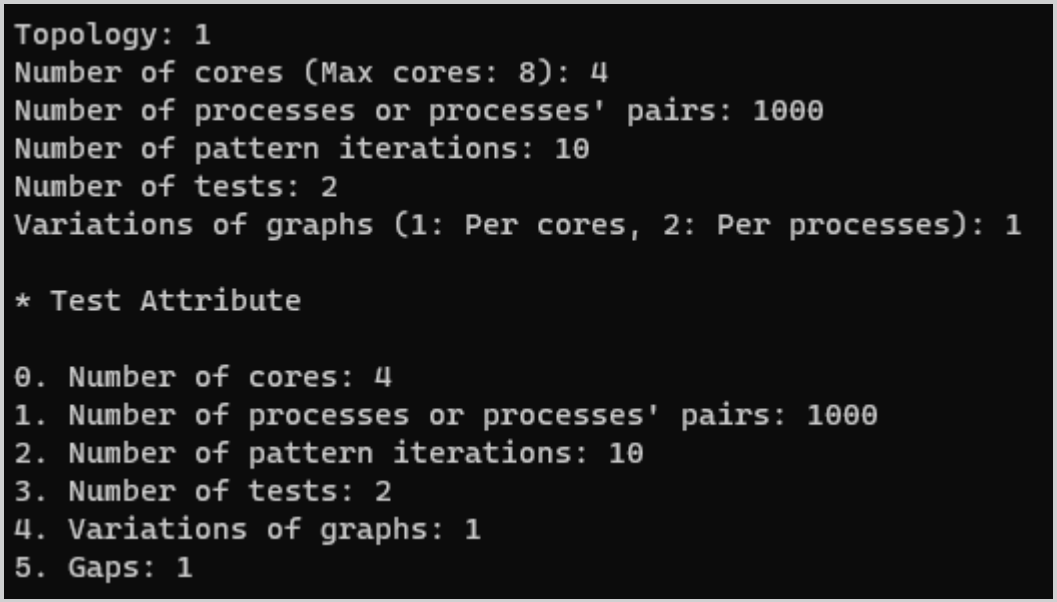
System Test – Python (Result)

Case No.	Test Result
2	
	입력한 Type에 대한 선택지로 정상적으로 진행되는 것을 확인하였습니다.

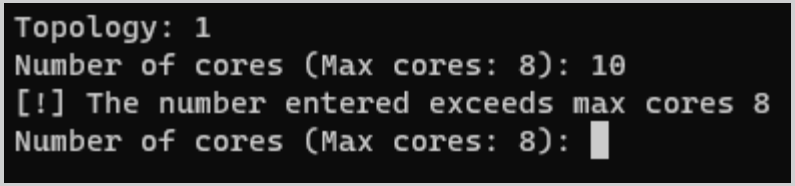
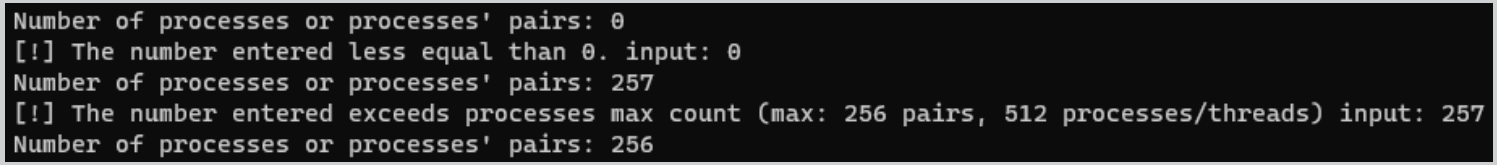
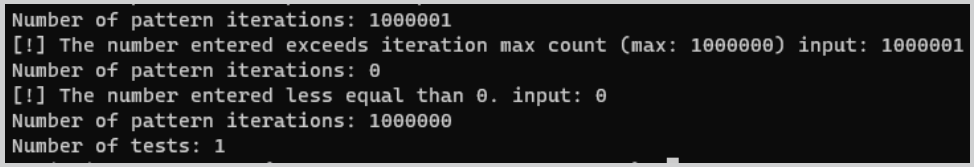
System Test – Python (Result)

Case No.	Test Result
3	
	Topology에 해당하는 값을 입력할 경우 정상적으로 다음단계에 진입하는 것을 확인하였습니다.
4	
	Topology에 해당하지 않는 값을 입력할 경우 오류 메시지를 출력합니다.

System Test – Python (Result)

Case No.	Test Result
5	
	사용자가 입력한 Test 속성 값이 올바르게 저장되었는지 확인하였습니다.

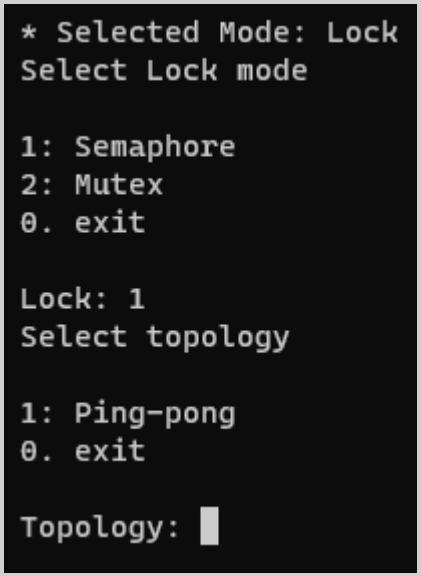
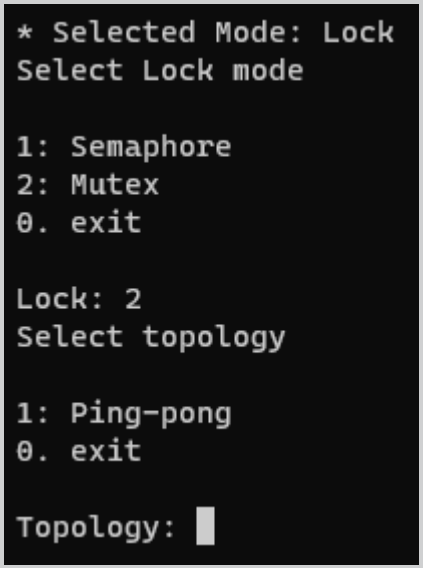
System Test – Python (Result)

Case No.	Test Result
6	 시스템이 가진 코어수를 초과한 값을 입력할 경우 오류메시지를 출력하는 것을 확인하였습니다.
7	 사용자가 입력한 pair의 2배를 한 값이 Process/Thread의 최대 개수를 초과하면 오류메시지를 출력하는 것을 확인하였습니다.
8	 실제 Testing은 C언어로 작성된 실행파일에서 진행되므로 C언어에서의 최대정수값 (unsigned __int64)의 값을 테스트 반복 횟수값의 범위로 정의하였고 해당 값을 초과할 경우 오류메시지를 출력하는 것을 확인하였습니다.

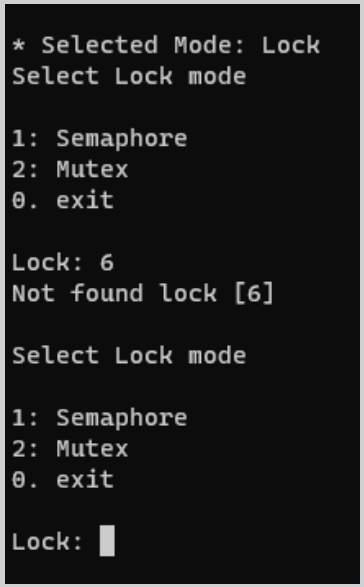
System Test - Python

Case No.	Req No.	Use case	Test description	Test Cases
9	3.2.1.4.	Select Lock Mode	선택된 락 모드의 입력창으로 변경이 되는지	1 ~ 2 범위 내의 정수 값
10			선택지에 존재하지 않는 값을 입력시 오류메시지를 출력하는지	1 ~ 2 범위 외의 정수 값
11	3.2.1.5.	Execute Whole Tests	그래프의 x축 값이 정상적으로 변경되면서 각 싱글 테스트를 수행하는지	범위 내의 정수 값
12			각 싱글 테스트를 수행한 시간 값들이 정상적으로 저장되는지	범위 내의 정수 값
13	3.2.1.6.	Execute Single Test	입력한 토폴로지 값, 프로세스 개수 값, 테스트 반복 횟수 값, 코어 개수 값이 변수에 올바르게 입력 되는지	범위 내의 정수 값
14			싱글 테스트에 대한 시간 측정값을 리턴하는지	범위 내의 정수 값
15	3.2.1.7.	Make a CSV File	테스팅 결과 값을 csv 파일을 통해 정상적으로 생성하는지	범위 내의 정수 값
16	3.2.1.8.	Print Graph	Y를 입력 받았을때 그래프를 생성하는지	Y
17			Y이외의 값을 입력 받았을때 그래프를 생성하지 않는지	Y가 아닌 값

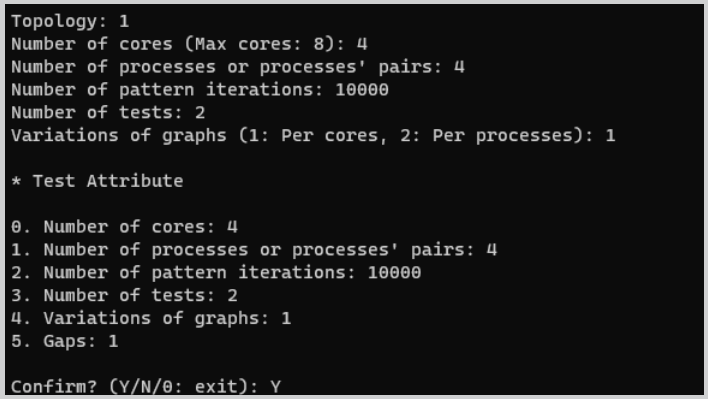
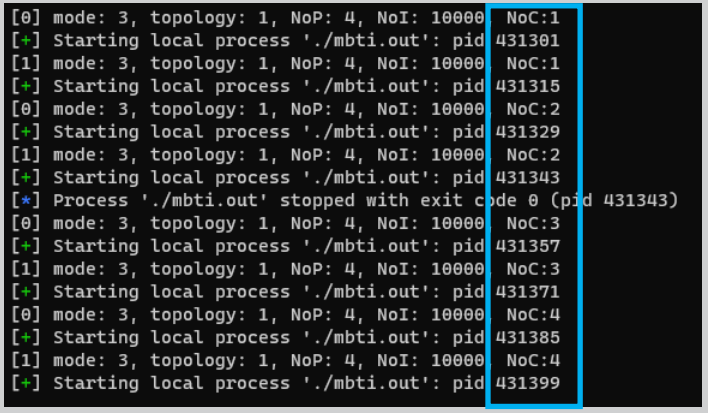
System Test – Python (Result)

Case No.	Test Result
9	 
	Lock 모드 중에서 1 ~ 2의 숫자를 입력할 경우 정상적으로 다음 단계에 진입하는 것을 확인하였습니다.

System Test – Python (Result)

Case No.	Test Result
10	
	Lock 모드 중에서 1 ~ 2 이외의 숫자를 입력할 경우 오류메시지를 출력하는 것을 확인하였습니다.

System Test – Python (Result)

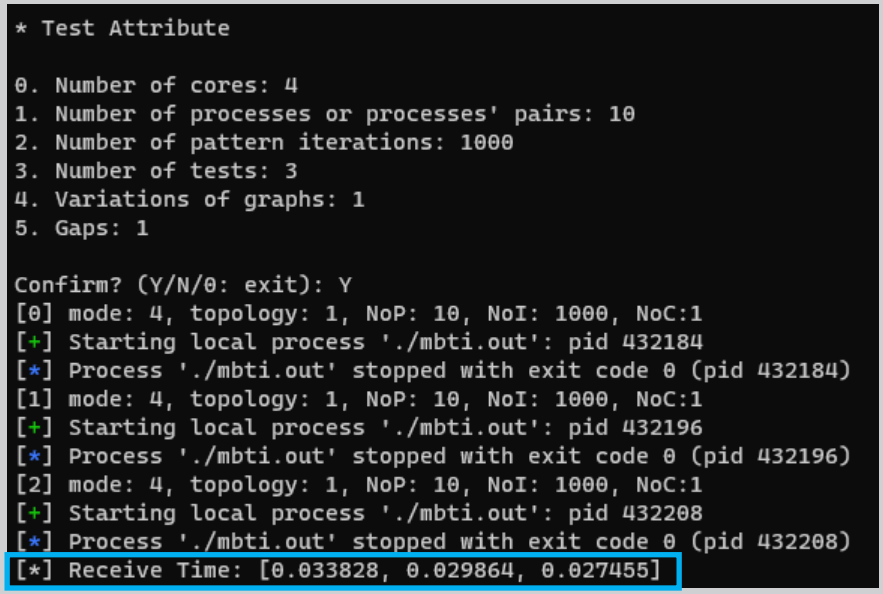
Case No.	Test Result
11	 

```
Topology: 1
Number of cores (Max cores: 8): 4
Number of processes or processes' pairs: 4
Number of pattern iterations: 10000
Number of tests: 2
Variations of graphs (1: Per cores, 2: Per processes): 1

* Test Attribute

0. Number of cores: 4
1. Number of processes or processes' pairs: 4
2. Number of pattern iterations: 10000
3. Number of tests: 2
4. Variations of graphs: 1
5. Gaps: 1

Confirm? (Y/N/0: exit): Y
```

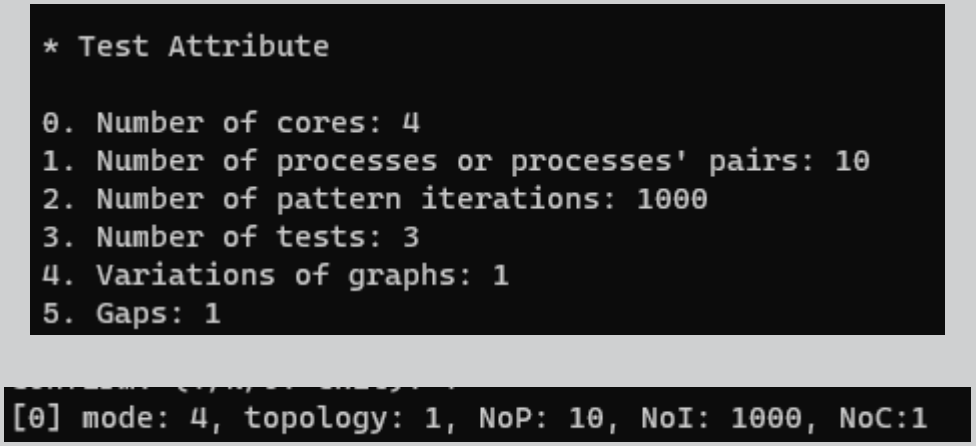
```
[0] mode: 3, topology: 1, NoP: 4, NoI: 10000, NoC:1
[+] Starting local process './mbti.out': pid 431301
[1] mode: 3, topology: 1, NoP: 4, NoI: 10000, NoC:1
[+] Starting local process './mbti.out': pid 431315
[0] mode: 3, topology: 1, NoP: 4, NoI: 10000, NoC:2
[+] Starting local process './mbti.out': pid 431329
[1] mode: 3, topology: 1, NoP: 4, NoI: 10000, NoC:2
[+] Starting local process './mbti.out': pid 431343
[*] Process './mbti.out' stopped with exit code 0 (pid 431343)
[0] mode: 3, topology: 1, NoP: 4, NoI: 10000, NoC:3
[+] Starting local process './mbti.out': pid 431357
[1] mode: 3, topology: 1, NoP: 4, NoI: 10000, NoC:3
[+] Starting local process './mbti.out': pid 431371
[0] mode: 3, topology: 1, NoP: 4, NoI: 10000, NoC:4
[+] Starting local process './mbti.out': pid 431385
[1] mode: 3, topology: 1, NoP: 4, NoI: 10000, NoC:4
[+] Starting local process './mbti.out': pid 431399
```

현 테스트의 경우는 Core 수를 증가시키면서 테스트 하는 과정을 출력한 것으로
NoC (Number of Cores) 가 증가하면서 각 싱글 테스트가 진행되는 것을 확인 하였습니다.

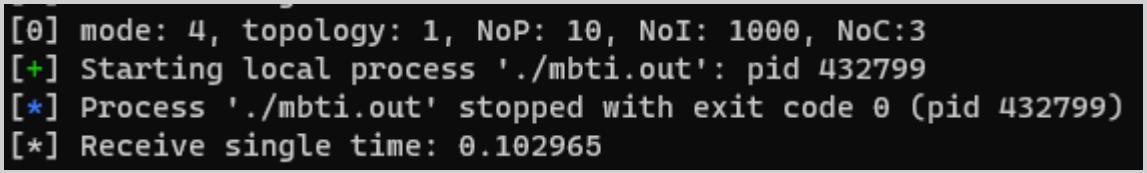
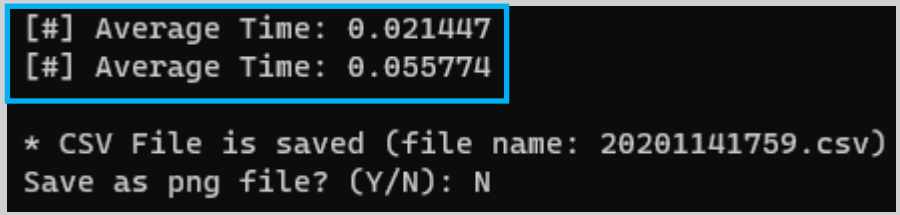
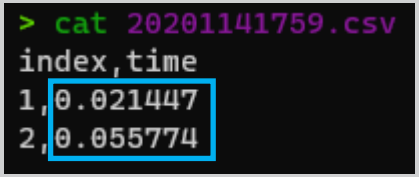
System Test – Python (Result)

Case No.	Test Result
12	
	각 싱글 테스트를 수행한 시간들이 정상적으로 반환되어 저장되는 것을 리스트로 확인하였습니다.

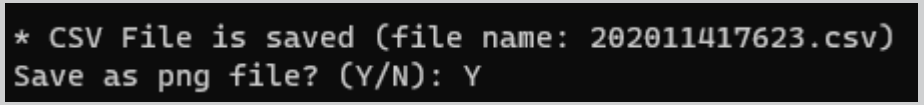
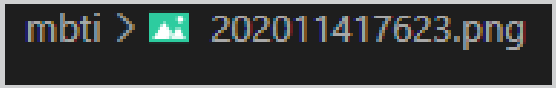
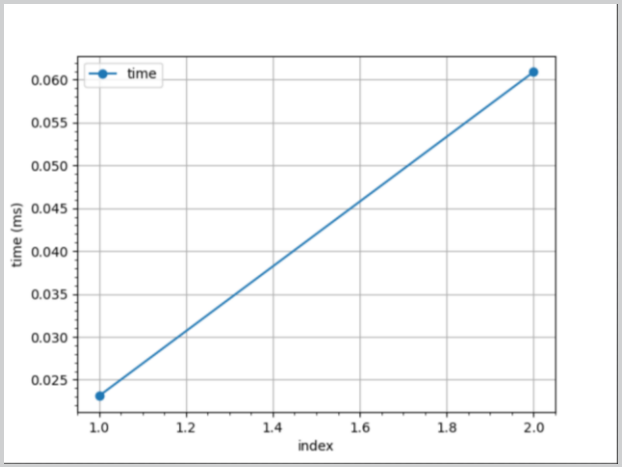
System Test – Python (Result)

Case No.	Test Result
13	
	사용자가 입력한 토폴로지 값, 프로세스 개수 값, 테스트 반복 횟수 값, 코어 개수 값이 정상적으로 출력되는 것을 확인하였습니다.

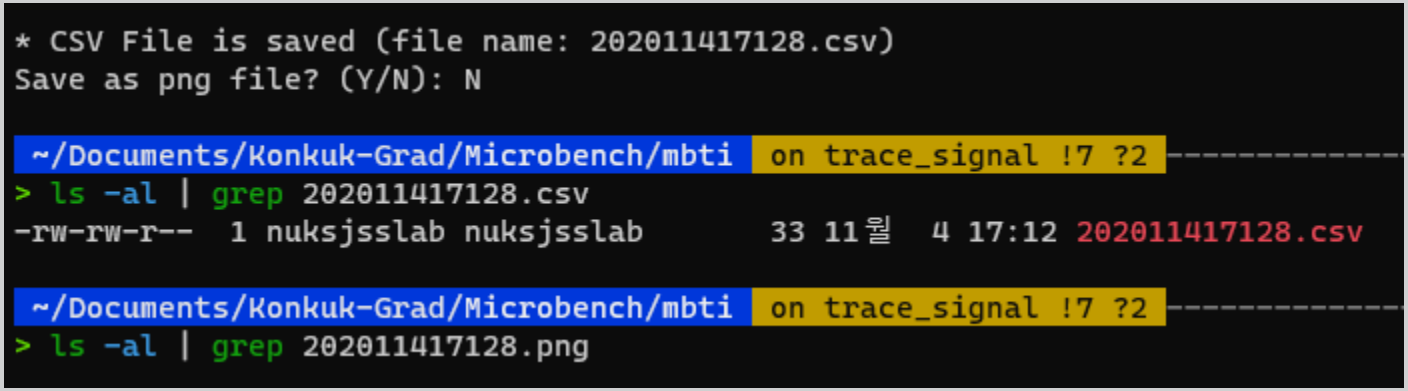
System Test – Python (Result)

Case No.	Test Result
14	
	하나의 싱글 테스트는 하나의 시간 값을 갖게 되는데, 하나의 시간 값을 정상적으로 반환하는 것을 출력을 통하여 확인하였습니다.
15	 
	테스팅 결과 값을 출력하고 CSV 파일을 확인하여 같은 값들이 들어가 있는것을 확인하였습니다.

System Test – Python (Result)

Case No.	Test Result
16	   Y를 입력 받았을 시 그래프를 생성하는 것을 확인하였습니다.

System Test – Python (Result)

Case No.	Test Result
17	
	Y이외의 값을 입력 받았을 때 ls 명령을 사용하여 그래프 이미지가 생성되지 않았음을 확인하였습니다.

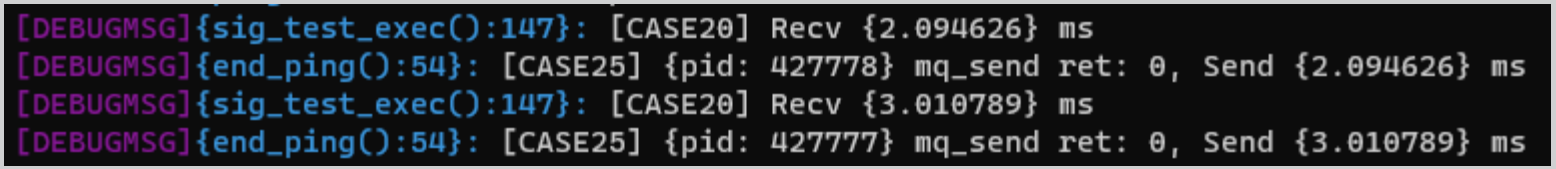
System Test - Signal

Case No.	Req No.	Use case	Test description	Test Cases
18	3.2.2.1.	Signal Test Main	사용자가 설정한 테스팅 설정값을 올바르게 가져오고 환경 초기화에 정상적인 값이 사용되는지 확인한다.	범위 내의 정수값
19	3.2.2.2.	Signal Test Initialization	테스팅을 위한 Topology 초기화 함수를 실행하여 환경 초기화를 위한 값을 넘겨주고 정상적으로 프로세스들이 생성되었을 경우 pid리스트가 반환되고 그렇지 않을경우 null이 반환되는지 확인	범위 내의 정수값
20	3.2.2.3.	Signal Test Execution	테스팅을 위해 생성된 프로세스들을 실행시켜서 실행한 모든 프로세스로 부터 시간값을 받아오고 결과값으로 변환하여 반환하는지	범위 내의 정수값
21	3.2.2.4.1.	init_pingpong	자식 process들을 정상적으로 생성하고 자식process들의 pid를 반환하는지?	범위 내의 정수값
22			생성된 자식 프로세스들은 부모프로세스를 받기 전까지 대기 상태인지?	범위 내의 정수값

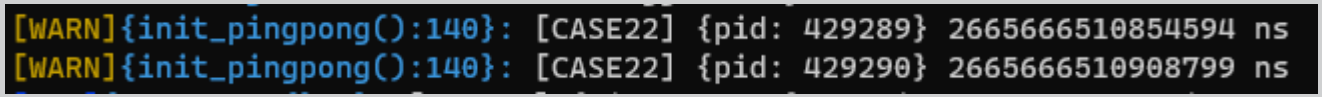
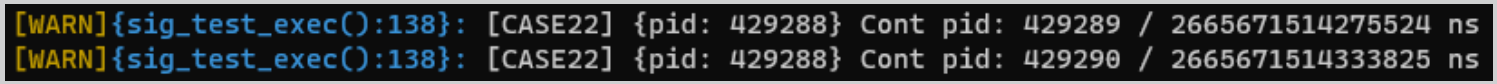
System Test – Signal (Result)

Case No.	Test Result	
18	`> ./mbti.out 1 1 2 10 1` `[DEBUGMSG]{sig_test():24}: [CASE18] topology: 1, Number of processes' pairs: 2, iter: 10, cores: 1`	
	mbti.out 바로 다음의 1은 signal이 므로 제외하고 나머지 값들은 정상적으로 출력합니다.	
19	`[DEBUGMSG]{sig_test_init():69}: [CASE19] pid_list: 427777 427778`	`} else if(pid_arr[i] < 0) { // Fork Error` `  PRINTERROR("[CASE19] Fork Failed!\n");` `  for(int j = 0; j < i; j++){` `  kill(pid_arr[i], SIGKILL);` `  }` `  free(pid_arr);` `  return NULL;`
	자식 Process 들의 PID를 정상적으로 받아온 것을 확인하였습니다. (Case 21과 같이 확인)	중간에 Fork가 실패할 경우 NULL을 Return 합니다.

System Test – Signal (Result)

Case No.	Test Result
20	
	시간값을 정상적으로 받아오는 것을 확인 (Case 25와 같이 확인)
21	
	생성된 자식 프로세스의 pid 리스트를 반환받습니다.

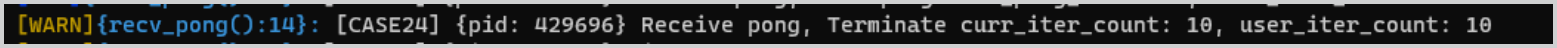
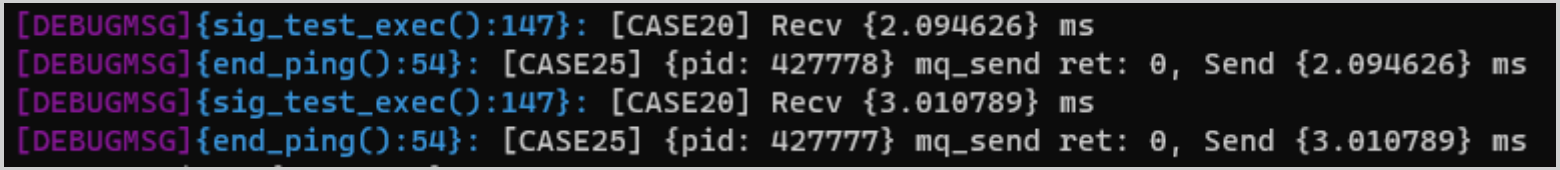
System Test – Signal (Result)

Case No.	Test Result
22	 Fork 하고 부모의 Signal을 기다림  자식은 Fork 하고 부모를 기다리고 있고 부모가 자식을 Triggering 한 시간 (5초) 을 측정한 결과로 PID 429289: 2665671514275524 - 2665666510854594 = 5003420930 ns = 5.003 sec PID 429290: 2665671514333825 - 2665666510908799 = 5003425026 ns = 5.003 sec 위와 같이 원하는 사용자가 원하는 시간에 모든 Ping 프로세스가 시작한다는 것을 확인.

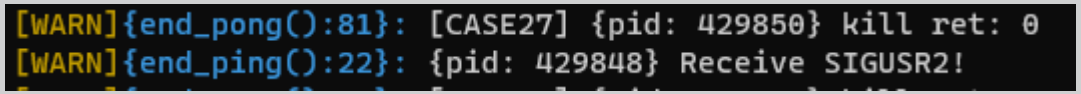
System Test - Signal

Case No.	Req No.	Use case	Test description	Test Cases
23	3.2.2.4.2.	recv_pong	갱신된 pong의 값이 pattern 반복횟수보다 작을 경우 pong process에 ping signal을 전송하는지?	실행 될 경우
24			갱신된 pong의 값이 pattern 반복횟수보다 같을 경우 pong process에 종료 signal을 전송하는지?	실행 될 경우
25	3.2.2.4.3.	end_ping	메시지큐에 측정된 시간값을 정상적으로 전송하는지?	실행 될 경우
26	3.2.2.4.4.	recv_ping	갱신된 ping의 값이 pattern 반복횟수보다 작을 경우 ping process에 pong signal을 전송하는지?	실행 될 경우
27	3.2.2.4.5.	end_pong	ping process에 종료 시그널을 보내는지?	실행 될 경우

System Test – Signal (Result)

Case No.	Test Result
23	
	현재 반복 횟수가 사용자가 설정한 반복횟수보다 작을 경우 출력됨.
24	
	현재 반복 횟수가 사용자가 설정한 반복횟수와 같을 경우 종료를 위한 분기로 들어가고 출력됨.
25	
	출력 버퍼의 순서에 의하여 역순으로 출력되었으나 각 end_ping 함수에서 정상적으로 시간값을 보내는 것을 확인할 수 있었음.

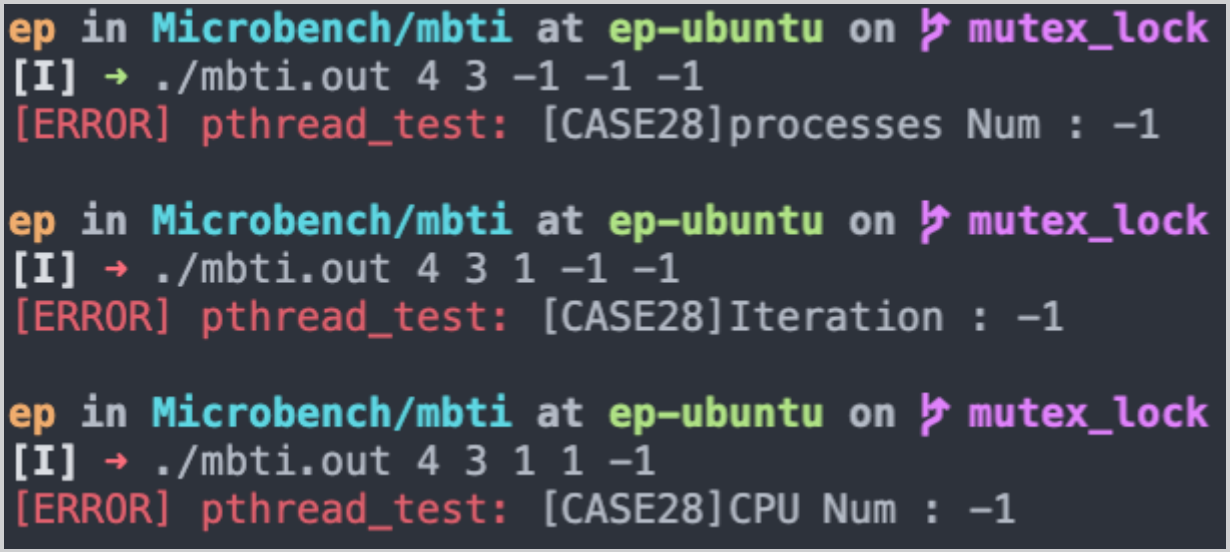
System Test – Signal (Result)

Case No.	Test Result
26	
	현재 받은 ping의 수가 사용자가 설정한 반복횟수보다 작을 경우 출력됨.
27	
	다른 Process에 Signal을 보내는 kill 함수의 Return 값을 확인하여 kill이 정상적으로 종료한 것을 확인하였고, SIGUSR2의 Handler인 end_ping 함수가 정상적으로 실행됨을 확인하였습니다.

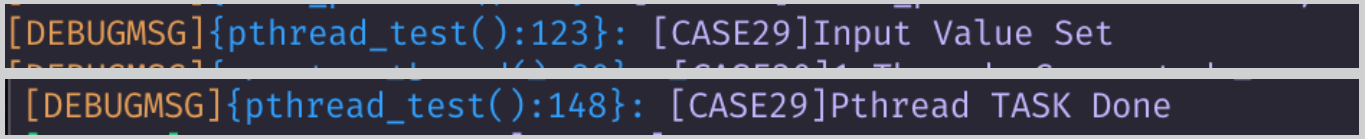
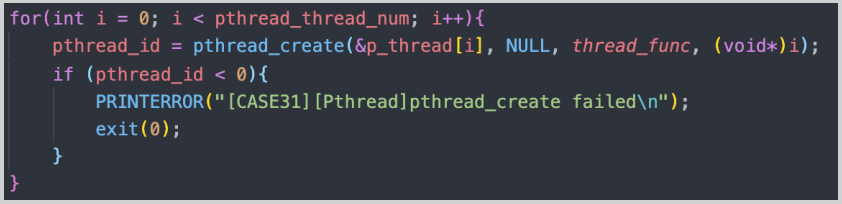
System Test - Mutex

Case No.	Req No.	Use case	Test description	Test Cases
28	3.2.3.1.	pthread_test	성능측정을 위해 생성된 timespec이 할당 해제되지 못하면 오류 메시지를 표시하는지	범위 밖의 정수 값
29			정상적으로 Input Value를 반영하고 create_pthread를 호출하는지	범위 내의 정수 값
30	3.2.3.2.	create_pthread	정상적으로 Input Value 에서 받은 값만큼 thread 를 생성하는지	범위 내의 정수 값
31			시스템이 정상적으로 스레드를 생성하지 못하면 오류메시지를 출력하는지	범위 밖의 정수 값
32	3.2.3.3.	pthread_setaffinity	Input Value로 부터 받아온 값이 음수일 경우 오류메시지를 출력하는지	범위 밖의 정수 값
33			시스템이 성공적으로 Core affinity를 설정하지 못하면 오류 메시지를 출력하는지	범위 밖의 정수 값
34			시스템이 성공적으로 Core affinity를 설정하는지	범위 내의 정수 값
35	3.2.3.4.	init_pthread	성능측정을 위한 각 timespec 이 동적할당이 되는지	범위 내의 정수 값
36			성능측정을 위한 각 timespec 이 동적할당에 실패했을시 오류 메시지를 표시하는지	범위 밖의 정수 값
37	3.2.3.5.	exit_pthread	성능측정을 위해 생성된 timespec이 free 되는지	실행 시
38			성능측정을 위해 생성된 timespec이 할당 해제되지 못하면 오류 메시지를 표시하는지	실행 시

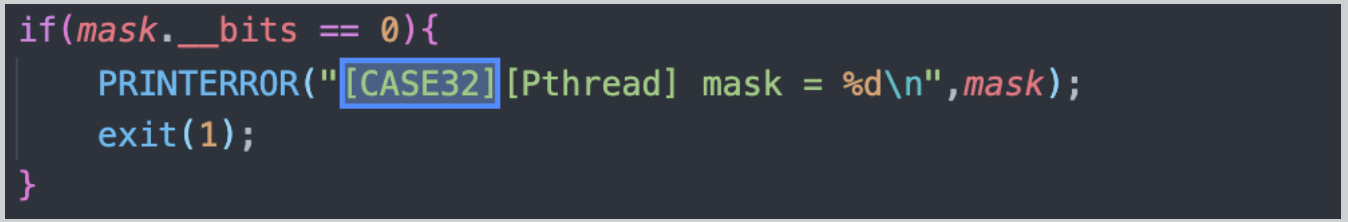
System Test – Mutex (Result)

Case No.	Test Result
28	
	Input Value로 부터 받아온 값이 음수일 경우 오류 메시지를 출력

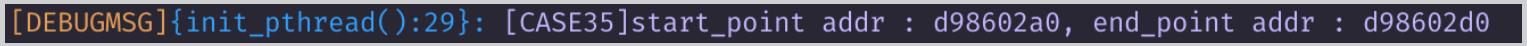
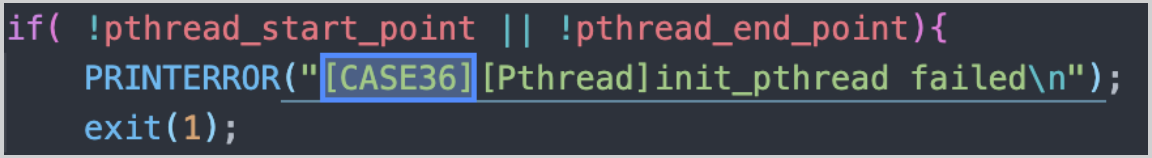
System Test – Mutex (Result)

Case No.	Test Result
29	
	정상적으로 Input Value를 반영하고 함수가 호출되었을 때 메시지를 출력
30	
	정상적으로 Input Value에서 받은 값만큼 thread를 생성시 메시지 출력
31	
	시스템이 정상적으로 스레드를 생성하지 못하면 오류메시지를 출력하도록 설정

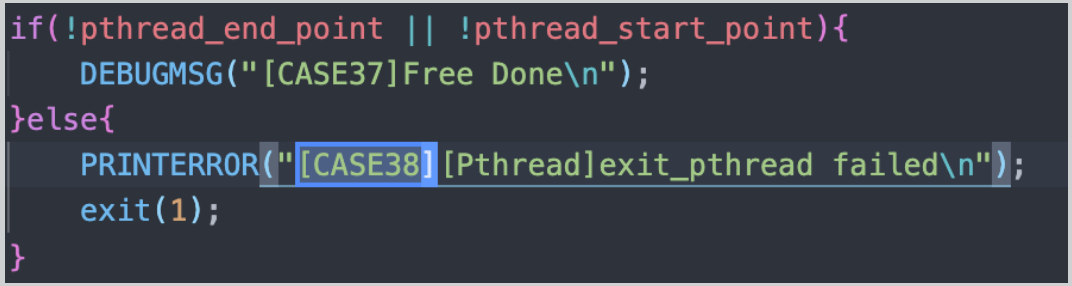
System Test – Mutex (Result)

Case No.	Test Result
32	 Input Value로 부터 받아온 값이 유효하지 않으면 오류메시지를 출력하도록 설정
33	 시스템이 성공적으로 Core affinity를 설정하지 못하면 오류 메시지를 출력하도록 설정

System Test – Mutex (Result)

Case No.	Test Result
34	
	시스템이 성공적으로 Core affinity를 설정하였을 시 메시지를 출력
35	
	성능측정을 위한 각 timespec이 동적 할당이 되었을 시 메시지를 출력
36	
	성능 측정을 위한 각 timespec이 동적 할당에 실패했을시 오류 메시지를 출력하도록 설정

System Test – Mutex (Result)

Case No.	Test Result
37	 성능 측정을 위해 생성된 timespec이 free되었을 시 메시지를 출력
38	 성능 측정을 위해 생성된 timespec이 할당 해제되지 못하면 오류 메시지를 출력하도록 설정

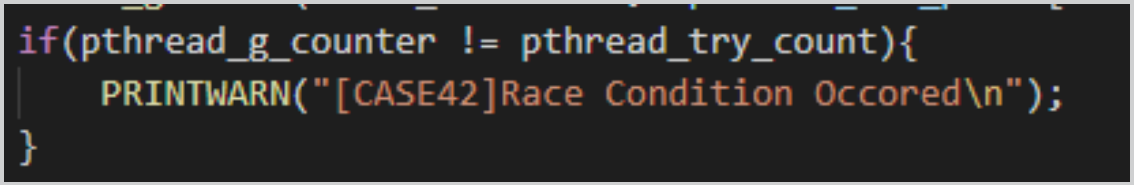
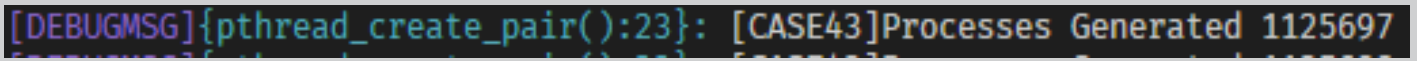
System Test - Mutex

Case No.	Req No.	Use case	Test description	Test Cases
39	3.2.3.6.	return_result	time_spec 구조체로부터 값을 받아와 스레드 혹은 프로세스 평균 소요시간을 산출하는지	실행 시
40	3.2.3.7.	pthread_global_ thread_act	스레드 마다 increase_counter를 정상적으로 실행하는지	실행 시
41			스레드마다 pthread_setaffinity를 정상적으로 호출하는지	실행 시
42	3.2.3.8.	increase_counter	글로벌 변수에 대하여 레이스 컨디션이 일어나는지	실행시
43	3.2.3.9.	pthread_create_pair	Input value 에 따라 process 가 생성되는지	범위 내의 정수 값
44			Input value 에 따라 process가 생성되지 않으면 메시지를 출력하는지	범위 밖의 정수 값
45	3.2.3.10.	pthread_spsc_thread_ act	정상적으로 pthread_producer와 pthread_consumer thread를 실행하는지	실행시
46	3.2.3.11.	pthread_pair1	임계영역에 값을 성공적으로 넣고 빼는지	실행시
47	3.2.3.12.	pthread_pair2	임계영역에 값을 성공적으로 넣고 빼는지	실행시

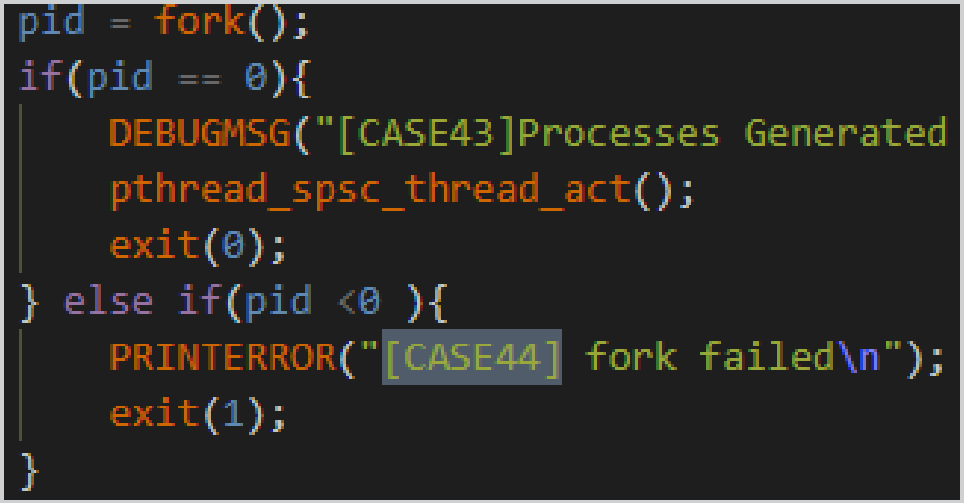
System Test – Mutex (Result)

Case No.	Test Result
39	
	Time_spec 구조체로부터 값을 받아와 스레드 혹은 프로세스의 평균 소요시간을 메시지로 출력
40	
	스레드 마다 increase_counter를 호출할 시 메시지로 출력
41	
	스레드마다 pthread_setaffinity를 정상적으로 호출하였을시 메시지로 출력

System Test – Mutex (Result)

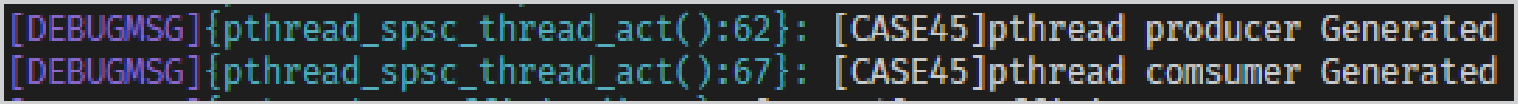
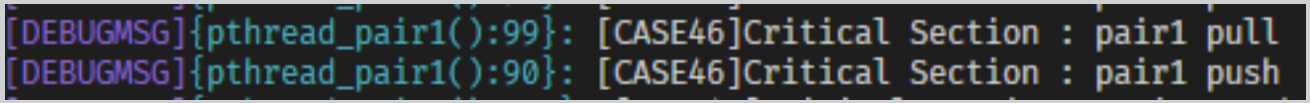
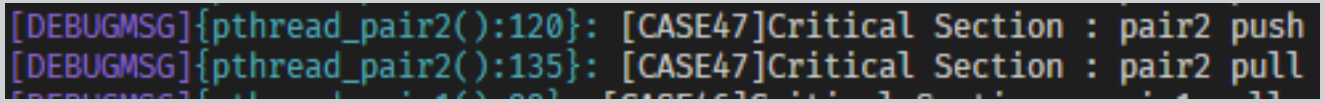
Case No.	Test Result
42	
	글로벌 변수에 대하여 레이스 컨디션이 발생시 오류메시지를 출력하도록 설정
43	
	Input Value에 따라 Process가 생성되는지 메시지로 출력

System Test – Mutex (Result)

Case No.	Test Result
44	
	Input Value 에 따라 프로세스 생성에 실패하였을 시 오류메시지를 출력하도록 설정

```
pid = fork();
if(pid == 0){
    DEBUGMSG("[CASE43]Processes Generated
    pthread_spsc_thread_act();
    exit(0);
} else if(pid <0 ){
    PRINTERROR("[CASE44] fork failed\n");
    exit(1);
}
```

System Test – Mutex (Result)

Case No.	Test Result
45	
	정상적으로 pthread_producer와 pthread_consumer thread를 실행하였을시 메시지로 출력
46	
	임계영역에 값을 성공적으로 넣고 가져올 때 메시지를 출력
47	
	임계영역에 값을 성공적으로 넣고 가져올 때 메시지를 출력

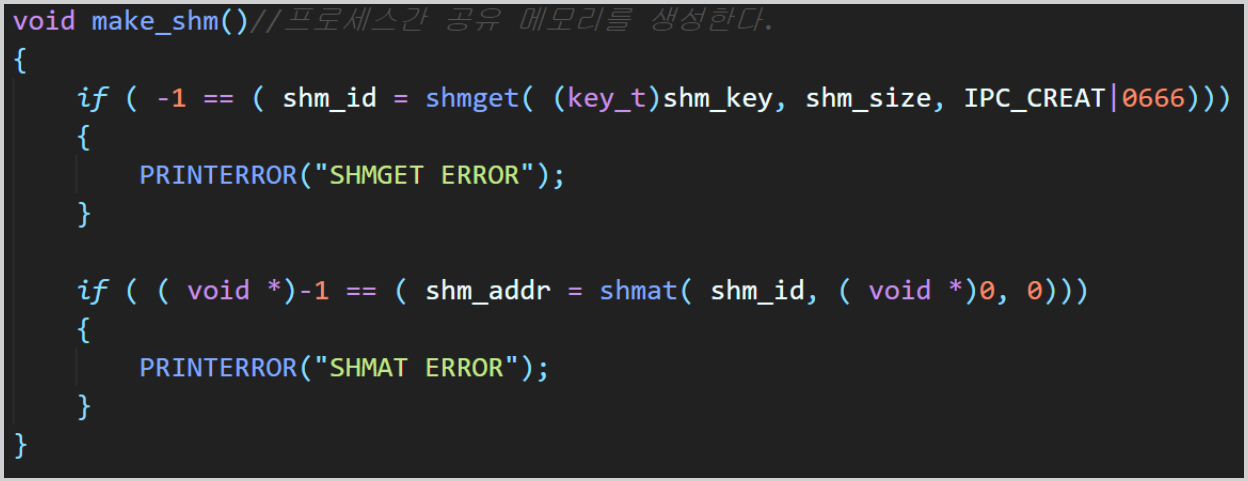
System Test - Semaphore

Case No.	Req No.	Use case	Test description	Test Cases
48	3.2.4.1.	Sem Producer	sem_put_item()을 성공적으로 호출하였는지	실행이 될 경우
49	3.2.4.2.	Sem Consumer	sem_consume_item()을 성공적으로 호출하였는지	실행이 될 경우
50	3.2.4.3.	Make Shm	프로세스들이 공유할 수 있는 메모리 영역이 생성되는 지	실행이 될 경우
51	3.2.4.4.	Sem Put Item	임계영역에 생산될 아이템을 넣었는지	실행이 될 경우
52	3.2.4.5.	Sem Consume Item	임계영역이 값을 삭제하였는지	실행이 될 경우
53	3.2.4.6.	Sem Set Core Affinities	사용자의 입력한 값 만큼의 core가 core set에 추가 되었는지	범위 내의 정수값

System Test – Semaphore (Result)

Case No.	Test Result
48	[DEBUGMSG]{/home/sean/Microbench/mbti/mbti_semaphore/mbti_sem_pp.c:5:sem_put_item()}: sem_put_item is called
	Sem_put_item()을 정상적으로 수행하는 것을 볼 수 있었습니다.
49	[DEBUGMSG]{/home/sean/Microbench/mbti/mbti_semaphore/mbti_sem_pp.c:10:sem_consume_item()}: sem_consume_item is called
	Sem_consume_item()을 정상적으로 수행하는 것을 볼 수 있었습니다.

System Test – Semaphore (Result)

Case No.	Test Result
50	
	PRINTERROR 메시지를 출력하지 않고 정상적으로 공용 메모리를 생성하는 것을 볼 수 있었습니다.

```
void make_shm()//프로세스간 공유 메모리를 생성한다.
{
    if ( -1 == ( shm_id = shmget( (key_t)shm_key, shm_size, IPC_CREAT|0666)))
    {
        PRINTERROR("SHMGET ERROR");
    }

    if ( ( void *)-1 == ( shm_addr = shmat( shm_id, ( void *)0, 0)))
    {
        PRINTERROR("SHMAT ERROR");
    }
}
```

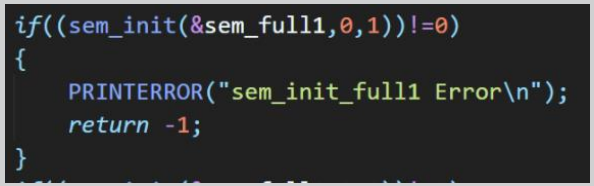
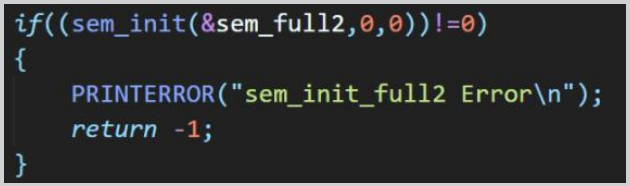
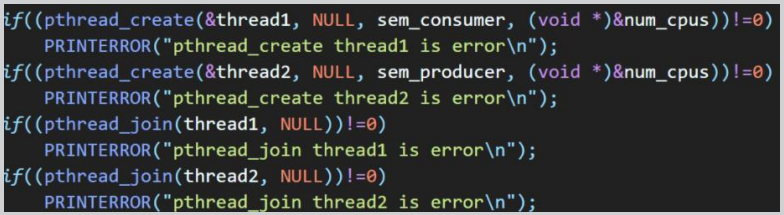
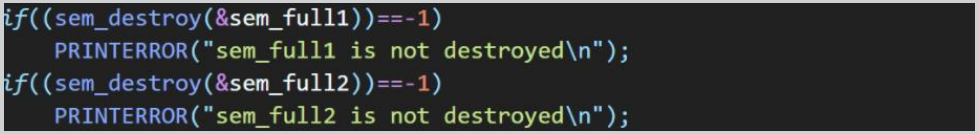
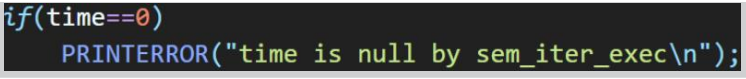
System Test – Semaphore (Result)

Case No.	Test Result
51	[DEBUGMSG]{/home/sean/Microbench/mbti/mbti_semaphore/mbti_sem_pp.c:5:sem_put_item()}: sem_put_item is called
	Sem_put_item()을 정상적으로 수행하는 것을 볼 수 있었습니다.
52	[DEBUGMSG]{/home/sean/Microbench/mbti/mbti_semaphore/mbti_sem_pp.c:10:sem_consume_item()}: sem_consume_item is called
	Sem_consume_item()을 정상적으로 수행하는 것을 볼 수 있었습니다.
53	[DEBUGMSG]{/home/sean/Microbench/mbti/mbti_semaphore/mbti_sem_pp.c:36:sem_set_core_affinities()}: CPU_SET 0 is added at cpuset
	정상적으로 core set에 코어를 추가하는 것을 볼 수 있었습니다.

System Test - Semaphore

Case No.	Req No.	Use case	Test description	Test Cases
54	3.2.4.7.	Sem Iter Exec	2개의 세마포어가 생성 및 초기화된다. 비정상 동작시 오류 메시지를 출력하는지.	실행이 되는 경우
55			사용자가 입력한 수만큼의 쓰레드를 생성하였는지. 생성하지 못하면 오류메시지를 출력하는지	범위 내의 정수 값
56			세마포어 변수를 제거하였는지	실행이 될 경우
57			최종 계산된 시간을 반환하는 지	범위 내의 정수 값
58	3.2.4.8.	Sem Make Processes	사용자가 입력한 수만큼의 프로세스를 생성하는지	범위 내의 정수 값
59			sem_iter_exec()함수를 실행하는 지	범위 내의 정수 값
60			공유 메모리에 각 시간 결과 값을 저장하는 지	범위 내의 정수 값
61			프로세스의 개수로 나눈 총 시간 값을 반환하는지	범위 내의 정수 값

System Test – Semaphore (Result)

Case No.	Test Result
54	 ```if((sem_init(&sem_full1,0,1))!=0) { PRINTERROR("sem_init_full1 Error\n"); return -1; }```  ```if((sem_init(&sem_full2,0,0))!=0) { PRINTERROR("sem_init_full2 Error\n"); return -1; }``` 세마포어 변수 2개를 초기화할 때, 오류를 반환하지 않고 정상적으로 수행됨을 볼 수 있었습니다.
55	 ```if((pthread_create(&thread1, NULL, sem_consumer, (void *)&num_cpus))!=0) PRINTERROR("pthread_create thread1 is error\n"); if((pthread_create(&thread2, NULL, sem_producer, (void *)&num_cpus))!=0) PRINTERROR("pthread_create thread2 is error\n"); if((pthread_join(thread1, NULL))!=0) PRINTERROR("pthread_join thread1 is error\n"); if((pthread_join(thread2, NULL))!=0) PRINTERROR("pthread_join thread2 is error\n");``` 쓰레드의 생성과 조인을 정상적으로 수행하고 오류를 반환하지 않음을 볼 수 있었습니다.
56	 ```if((sem_destroy(&sem_full1))==-1) PRINTERROR("sem_full1 is not destroyed\n"); if((sem_destroy(&sem_full2))==-1) PRINTERROR("sem_full2 is not destroyed\n");``` 세마포어 변수를 오류를 반환하지 않고 정상적으로 제거함을 볼 수 있었습니다.
57	 ```if(time==0) PRINTERROR("time is null by sem_iter_exec\n");``` 시간 값이 무의미한 값이 아닌 것을 오류를 반환하지 않음으로 볼 수 있었습니다.

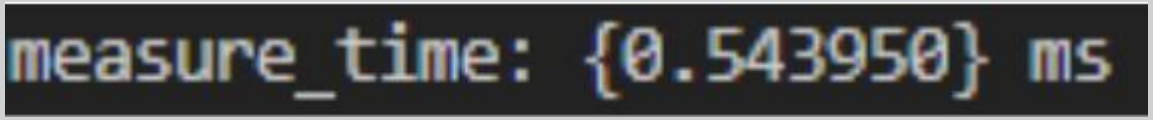
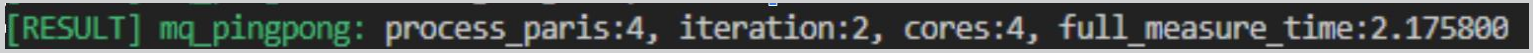
System Test – Semaphore (Result)

Case No.	Test Result
58	`if(count!=processes)` `  PRINTERROR("processes are not createad! They must be created %d more\n",(processes-count));` 원하는 수만큼의 프로세스의 수를 오류를 반환하지 않고 생성함을 알 수 있었습니다.
59	`[DEBUGMSG]{/home/sean/Microbench/mbti/mbti_semaphore/mbti_sem_pp.c:167:sem_make_processes()}: Wait() Child 1 END : statue NO0` 디버그 메시지를 통해 sem_iter_exec()을 수행하여 오류를 반환하지 않고 종료되어 sem_make_processes()에서 함수가 정상 종료됨을 볼 수 있었습니다.
60	`}else if(pid[i] == 0)` `{` `  time = (double*)shm_addr;` `  *(time+i) = sem_iter_exec(iter,num_cpus);` `  if((*(time+i))==0)` `  PRINTERROR("sem_iter_exec no.%d is not perfomed\n",i+1);` `  exit(0);` `}else` 각각의 시간 값을 받아와 오류를 반환하지 않고 공유 메모리인 time에 저장함을 볼 수 있었습니다.
61	`result = result/processes;` `if(result==0)` `  PRINTERROR("total result value error\n");` 총 결과 값을 받아와 정해진 프로세스의 수만큼 나눈 값을 정상적으로 출력함을 볼 수 있었습니다.

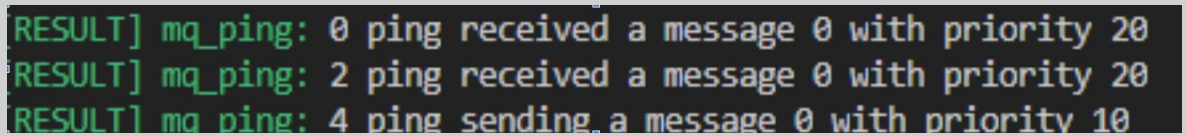
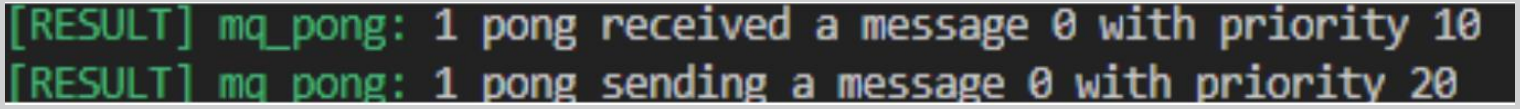
System Test - IPC

Case No.	Req No.	Use case	Test description	Test Cases
62	3.2.5.1.	IPC Test Execution	입력한 토폴로지의 함수를 사용자가 설정한 값으로 테스트한 평균 시간을 리턴하는지	범위 내의 정수 값
63	3.2.5.2.	IPC Test Execution	사용자가 설정한 값으로 핑퐁 테스트를 진행하고 실행된 시간을 리턴하는지	범위 내의 정수 값
64	3.2.5.3.	Set Core Affinity	시스템이 코어 어피니티가 사용자가 설정한 값으로 설정이 되는지	범위 내의 정수 값
65	3.2.5.4.	Message Queue Ping	사용자가 설정한 값으로 Pong 역할을 하는 프로세스와 메시지를 정상적으로 주고 받는지	범위 내의 정수 값
66			Pong 역할을 하는 프로세스와 메시지를 주고 받은 총 시간을 리턴하는지	범위 내의 정수 값
67	3.2.5.5.	Message Queue Pong	사용자가 설정한 값으로 Ping 역할을 하는 프로세스와 메시지를정상적으로 주고 받는지	범위 내의 정수 값
68	3.2.5.6	Measure Time	인자 값이 0인 경우 시간을 초기화하고 시간을 측정하는지	0
69			인자 값이 1인 경우 시간 측정을 종료하고 측정된 시간값을밀리초 단위로 리턴하는지	1

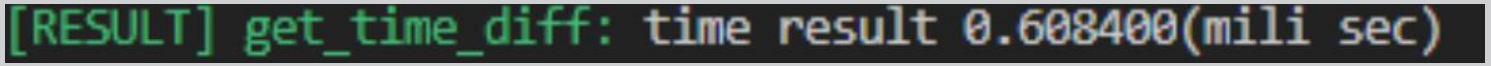
System Test – IPC (Result)

Case No.	Test Result
62	
	입력한 토폴로지의 함수를 사용자가 설정한 값으로 테스트한 평균 시간을 리턴하는것을 확인
63	
	사용자가 설정한 값으로 핑퐁 테스트를 진행하고 실행된 총 시간을 리턴하는것을 확인
64	
	시스템이 코어 어피니티가 사용자가 설정한 값으로 설정이 되는지 비트 단위로 출력하여 확인

System Test – IPC (Result)

Case No.	Test Result
65	
	사용자가 설정한 값으로 Pong 역할을 하는 프로세스와 메시지를 정상적으로 주고 받는지 확인
66	
	Pong 역할을 하는 프로세스와 메시지를 주고 받은 총 시간을 리턴하는지 확인
67	
	사용자가 설정한 값으로 Ping 역할을 하는 프로세스와 메시지를 정상적으로 주고 받는지 확인

System Test – IPC (Result)

Case No.	Test Result
68	
	인자 값이 0인 경우 시간을 초기화하고 시간을 측정하는지 확인
69	
	인자 값이 1인 경우 시간 측정을 종료하고 측정된 시간값을 밀리초 단위로 리턴하는지 확인

System Test Pass/Fail Criteria

Test No.	Req No.	Use case	Case Pass / Fail	Test No.	Req No.	Use case	Case Pass / Fail
1	3.2.1.1.	Select Mode	Case 1: **PASS** Case 2: **PASS**	9	3.2.1.8.	Print Graph	Case 16: **PASS** Case 27: **PASS**
2	3.2.1.2.	Select Topology	Case 3: **PASS** Case 4: **PASS**	10	3.2.2.1.	Signal Test Main	Case 18: **PASS**
3	3.2.1.3.	Input Value	Case 5: **PASS** / Case 6: **PASS** Case 7: **PASS** / Case 8: **PASS**	11	3.2.2.2.	Signal Test Initialization	Case 19: **PASS**
4	3.2.1.4.	Select Lock Mode	Case 9: **PASS** Case 10: **PASS**	12	3.2.2.3.	Signal Test Execution	Case 20: **PASS**
5	3.2.1.5.	Execute Whole Tests	Case 11: **PASS** Case 12: **PASS**	13	3.2.2.4.1.	init_pingpong	Case 21: **PASS** Case 22: **PASS**
6	3.2.1.6.	Execute Single Test	Case 13: **PASS** Case 14: **PASS**	14	3.2.2.4.2.	recv_pong	Case 23: **PASS** Case 24: **PASS**
7	3.2.1.7.	Make a CSV File	Case 15: **PASS**	15	3.2.2.4.3	end_ping	Case 25: **PASS**
8	3.2.1.8.	Print Graph	Case 16: **PASS** Case 27: **PASS**	16	3.2.2.4.4.	recv_pong	Case 26: **PASS**

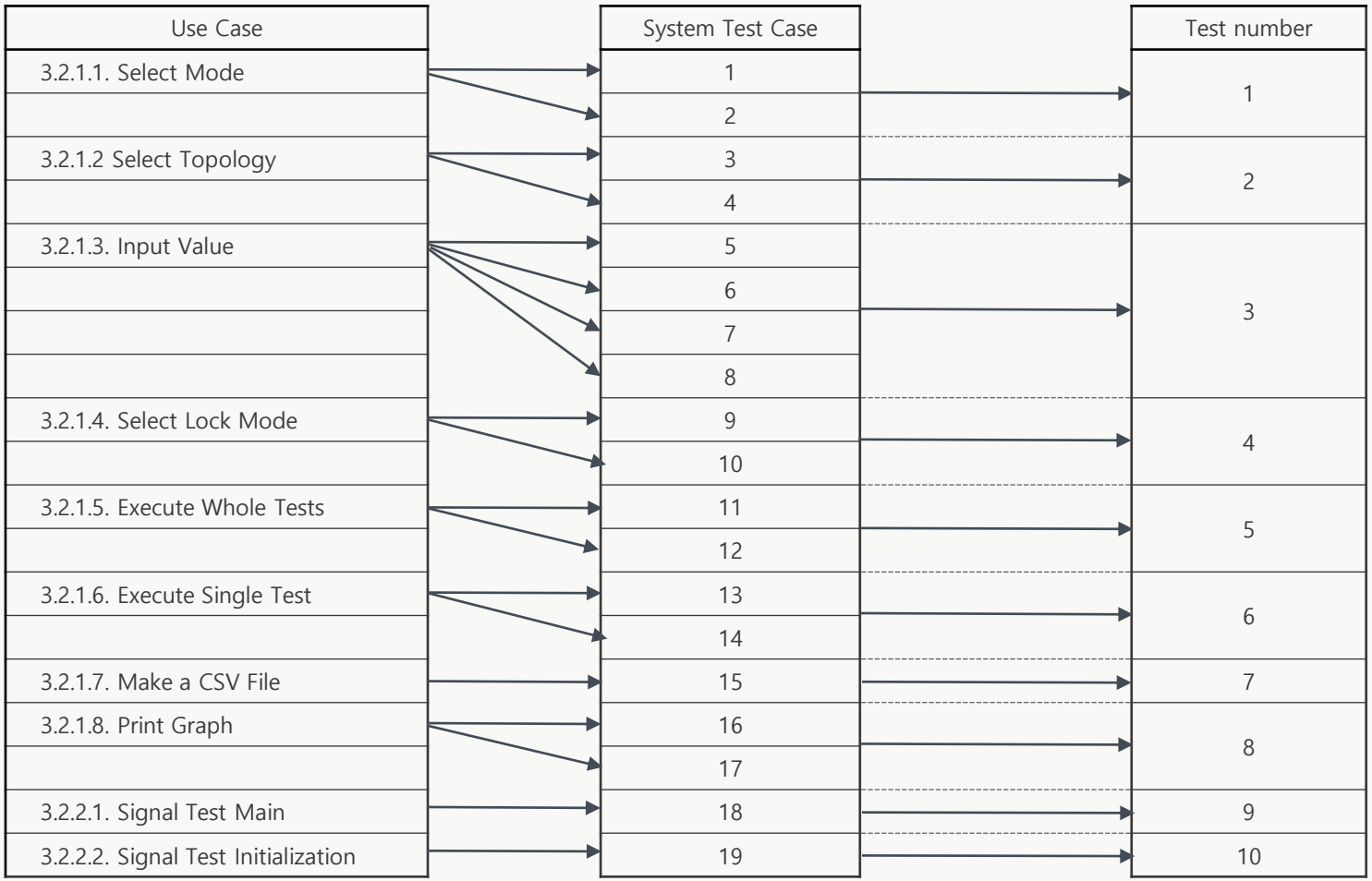
System Test Pass/Fail Criteria

Test No.	Req No.	Use case	Case Pass / Fail	Test No.	Req No.	Use case	Case Pass / Fail
17	3.2.3.1.	pthread_test	Case 28: **PASS** Case 29: **PASS**	25	3.2.3.9.	pthread_ create_pair	Case 43: **PASS** Case 44: **PASS**
18	3.2.3.2.	create_ pthread	Case 30: **PASS** Case 31: **PASS**	26	3.2.3.10.	pthread_spsc _thread_act	Case 45: **PASS**
19	3.2.3.3.	pthread_ sefaffinity	Case 32: **PASS** / Case 33: **PASS** Case 34: **PASS**	27	3.2.3.11.	pthread_ pair1	Case 46: **PASS**
20	3.2.3.4.	init_pthread	Case 35: **PASS** Case 36: **PASS**	28	3.2.3.12	pthread_ pair2	Case 47: **PASS**
21	3.2.3.5.	exit_pthread	Case 37: **PASS** Case 38: **PASS**	29	3.2.4.1.	Sem Producer	Case no.48 : **PASS**
22	3.2.3.6.	return_result	Case 39: **PASS**	30	3.2.4.2.	Sem Consumer	Case no.49 : **PASS**
23	3.2.3.7.	pthread_glob al_thread_act	Case 40: **PASS** Case 41: **PASS**	31	3.2.4.3.	Make Shm	Case no.50 : **PASS**
24	3.2.3.8.	increase_ counter	Case 42: **PASS**	32	3.2.4.4.	Sem Put Item	Case no.51 : **PASS**

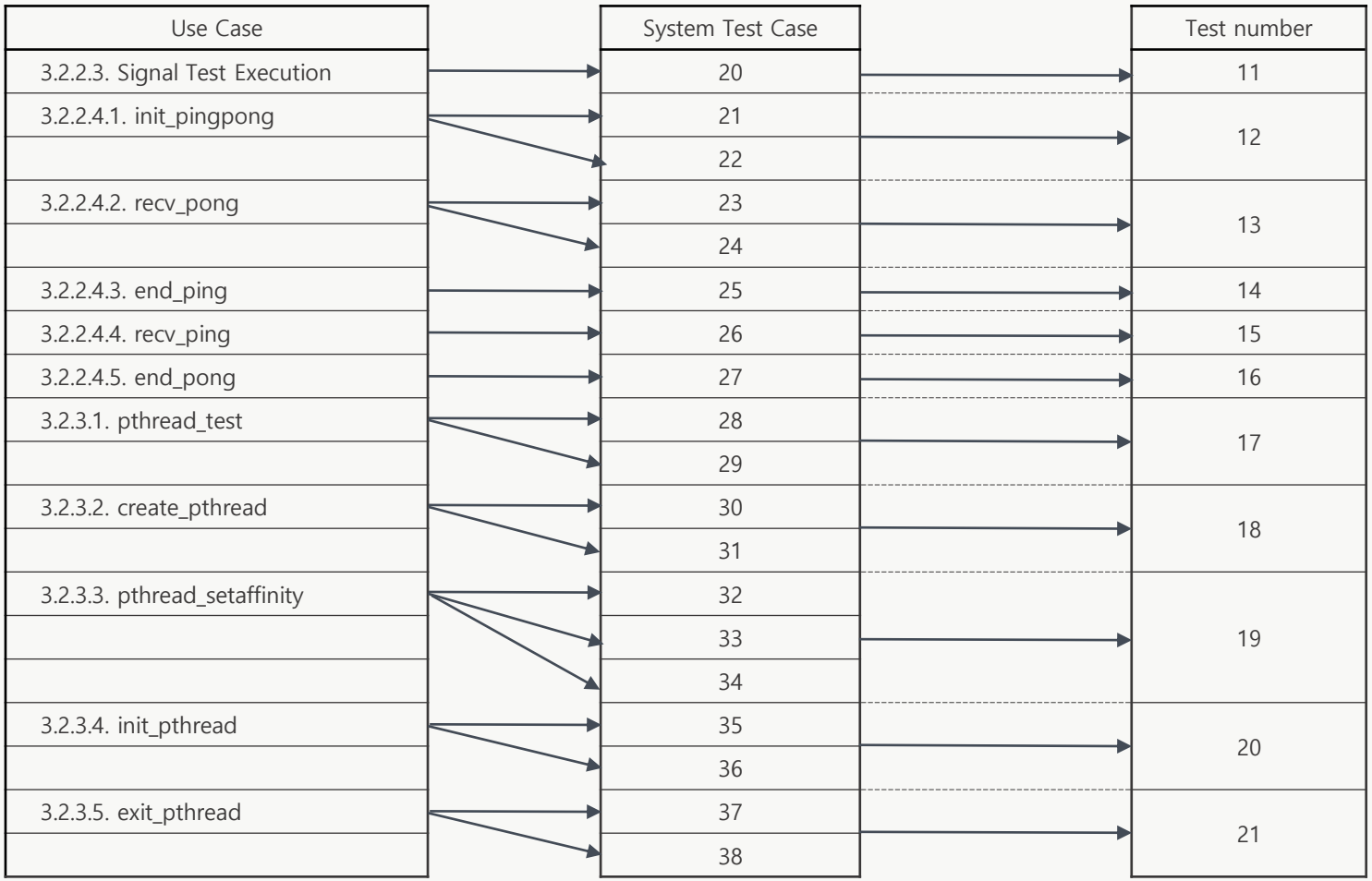
System Test Pass/Fail Criteria

Test No.	Req No.	Use case	Case Pass / Fail
33	3.2.4.5.	Sem Consume Item	Case no.52 : **PASS**
34	3.2.4.6.	Sem Set Core Affinities	Case no.53 : **PASS**
35	3.2.4.7.	Sem Iter Exec	Case no.54 : **PASS** Case no.55 : **PASS** Case no.56 : **PASS** Case no.57 : **PASS**
36	3.2.4.8	Sem Make Processes	Case no.58 : **PASS** Case no.59 : **PASS** Case no.60 : **PASS** Case no.61 : **PASS**
37	3.2.5.1	IPC Test Execution	Case 62: **PASS**
38	3.2.5.2	Message Queue Ping-Pong Test	Case 63: **PASS**
39	3.2.5.3	Set Core Affinity	Case 64: **PASS**
40	3.2.5.4	Message Queue Ping	Case 65: **PASS** Case 66: **PASS**
41	3.2.5.5	Message Queue Pong	Case 67: **PASS**
42	3.2.5.6	Measure Time	Case 68: **PASS** Case 69: **PASS**

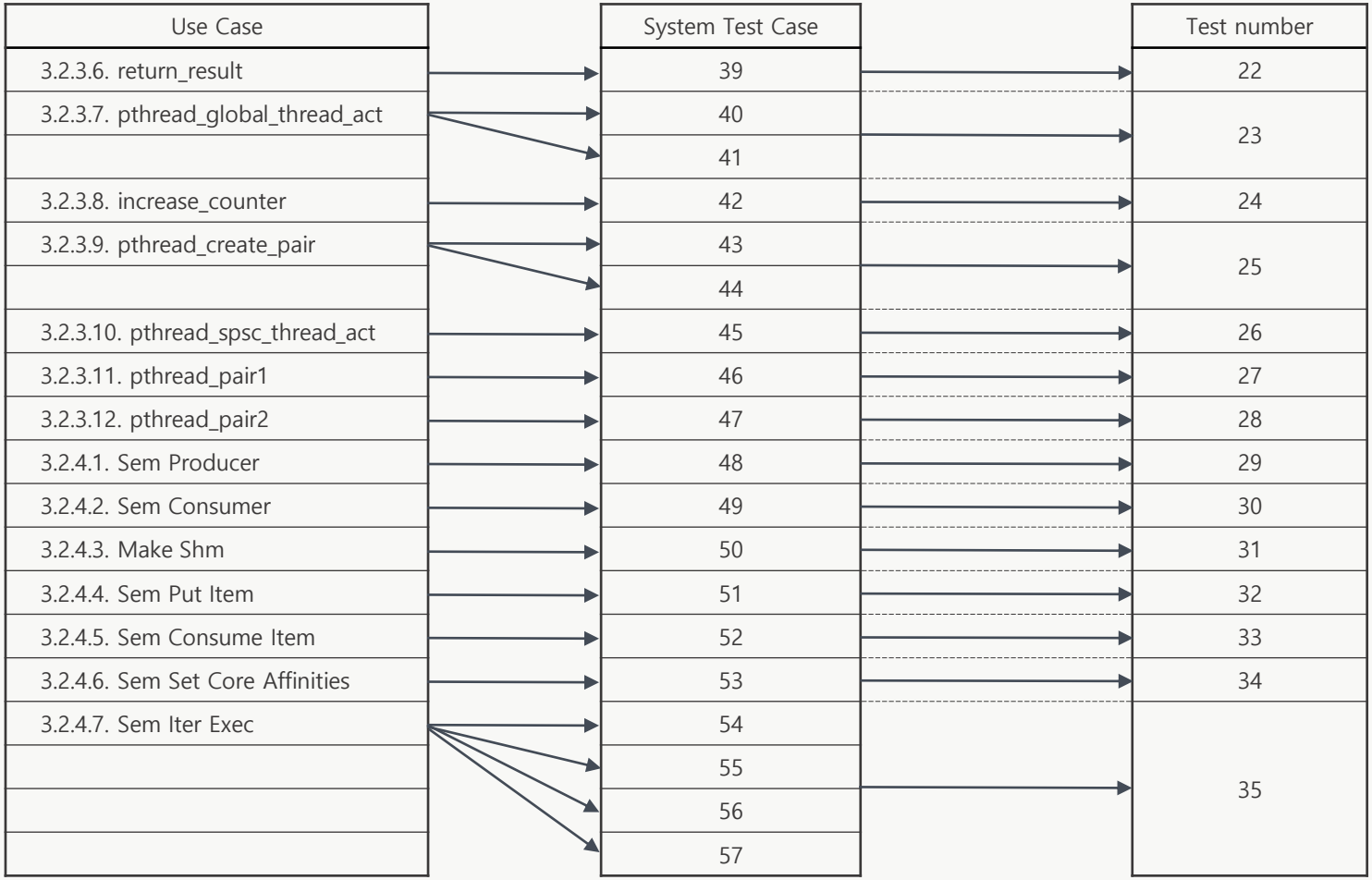
추적성 분석


Use Case
System Test Case
Test number
3.2.1.1. Select Mode
3.2.1.2 Select Topology
3.2.1.3. Input Value
3.2.1.4. Select Lock Mode
3.2.1.5. Execute Whole Tests
3.2.1.6. Execute Single Test
3.2.1.7. Make a CSV File
3.2.1.8. Print Graph
3.2.2.1. Signal Test Main
3.2.2.2. Signal Test Initialization
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
1
2
3
4
5
6
7
8
9
10

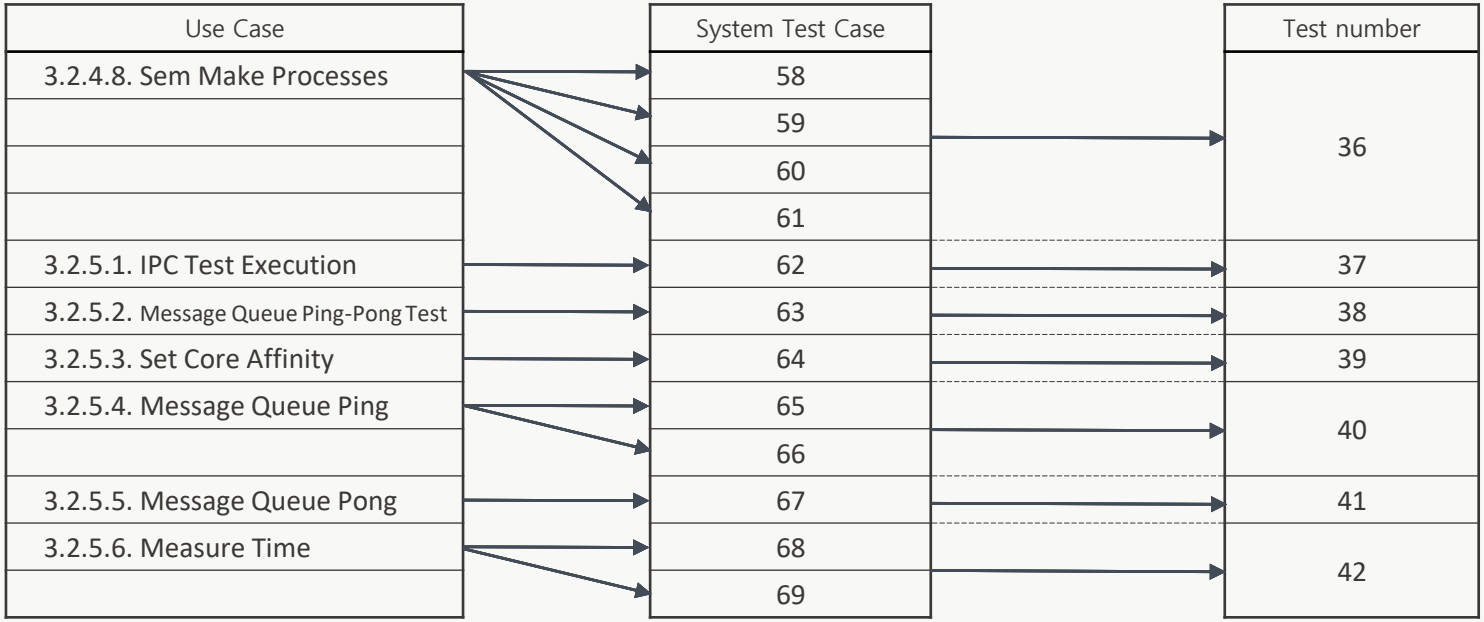
추적성 분석


Use Case
System Test Case
Test number
3.2.2.3. Signal Test Execution
3.2.2.4.1. init_pingpong
3.2.2.4.2. recv_pong
3.2.2.4.3. end_ping
3.2.2.4.4. recv_ping
3.2.2.4.5. end_pong
3.2.3.1. pthread_test
3.2.3.2. create_pthread
3.2.3.3. pthread_setaffinity
3.2.3.4. init_pthread
3.2.3.5. exit_pthread
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
11
12
13
14
15
16
17
18
19
20
21

추적성 분석


Use Case
3.2.3.6. return_result
3.2.3.7. pthread_global_thread_act
3.2.3.8. increase_counter
3.2.3.9. pthread_create_pair
3.2.3.10. pthread_spsc_thread_act
3.2.3.11. pthread_pair1
3.2.3.12. pthread_pair2
3.2.4.1. Sem Producer
3.2.4.2. Sem Consumer
3.2.4.3. Make Shm
3.2.4.4. Sem Put Item
3.2.4.5. Sem Consume Item
3.2.4.6. Sem Set Core Affinities
3.2.4.7. Sem Iter Exec
System Test Case
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
Test number
22
23
24
25
26
27
28
29
30
31
32
33
34
35

추적성 분석


Use Case
System Test Case
Test number
3.2.4.8. Sem Make Processes
58
59
60
61
36
3.2.5.1. IPC Test Execution
62
37
3.2.5.2. Message Queue Ping-Pong Test
63
38
3.2.5.3. Set Core Affinity
64
39
3.2.5.4. Message Queue Ping
65
66
40
3.2.5.5. Message Queue Pong
67
41
3.2.5.6. Measure Time
68
69
42

측정 시나리오

1. Increment Core Test

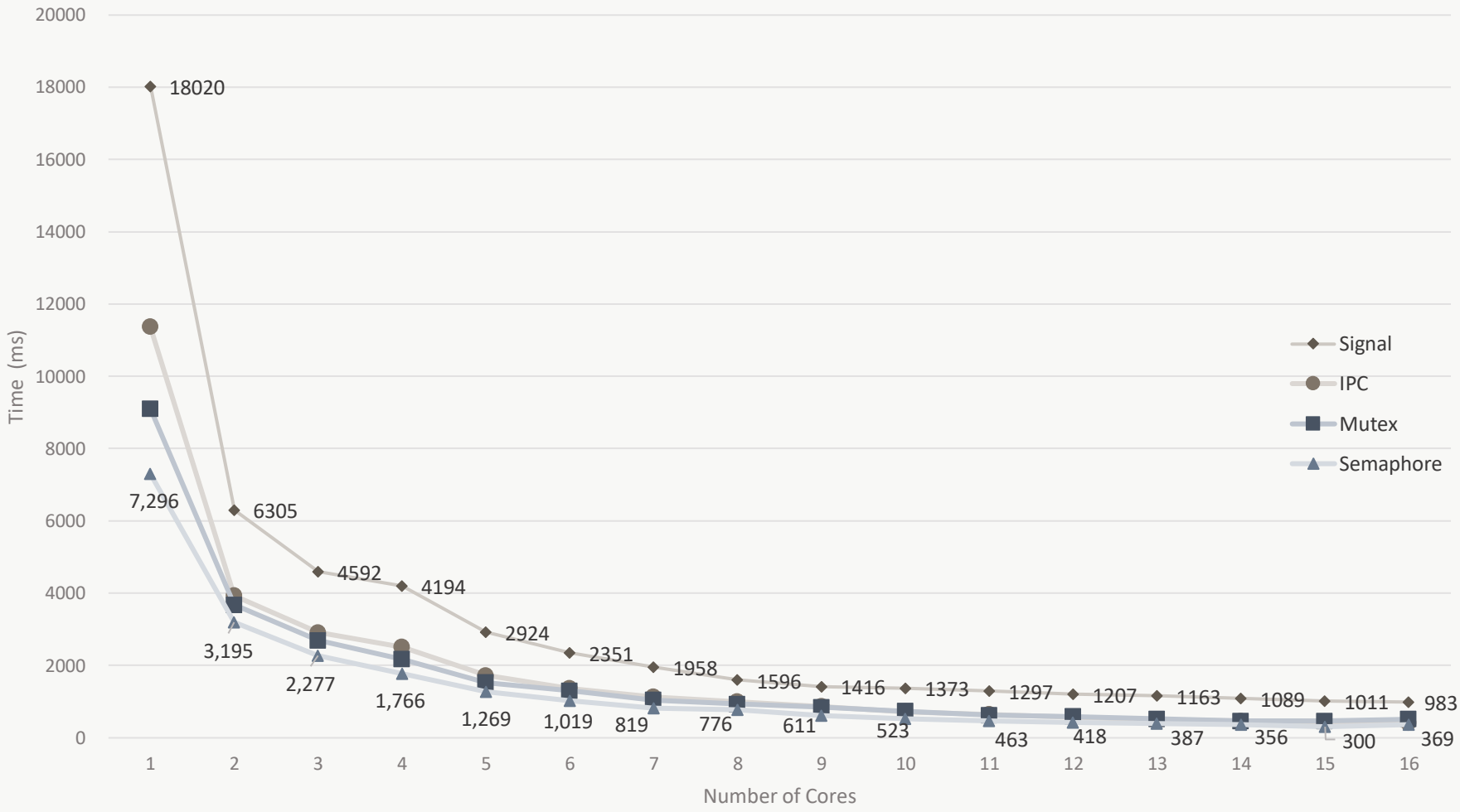
- 프로세스 수와 패턴 반복 수는 고정 시킨 채 코어 수만 증가시켜 결과를 분석해 볼 수 있다.

2. Increment Process Test

- 코어 수와 패턴 반복 수는 고정 시킨 채 프로세스 수만 증가시켜 결과를 분석해 볼 수 있다.

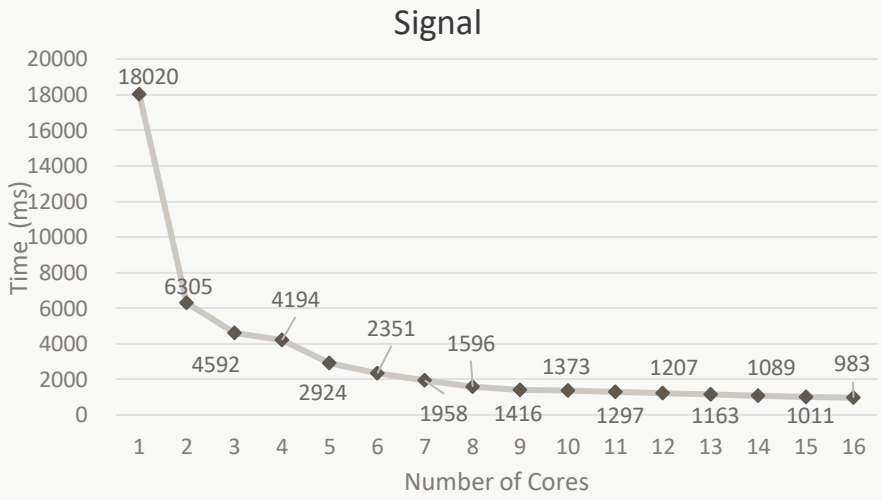
Increment Core Test

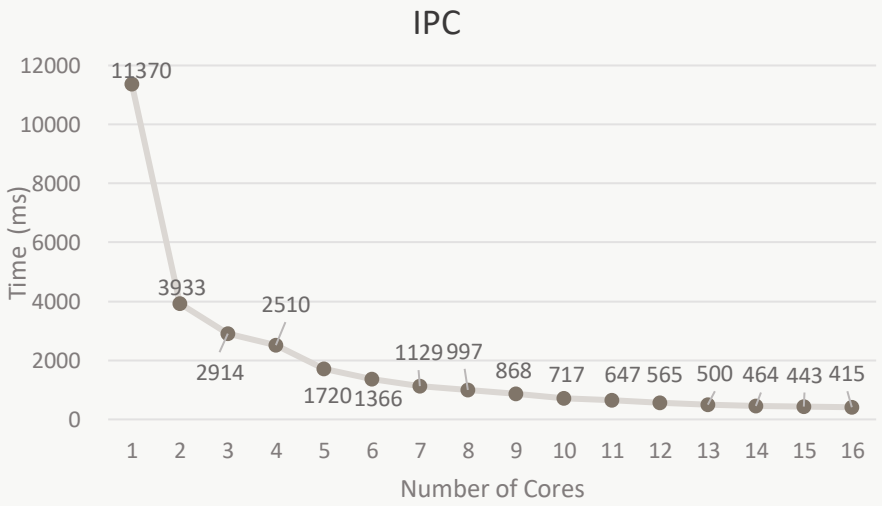
Execution Environment	
CPU	AMD Ryzen Threadripper 2950X
Cores	**Linear increment (1~16)**
Pairs	**256 pairs (512 processes or threads) [Fixed]**
Iterations	10,000
Unit	ms (milli second)

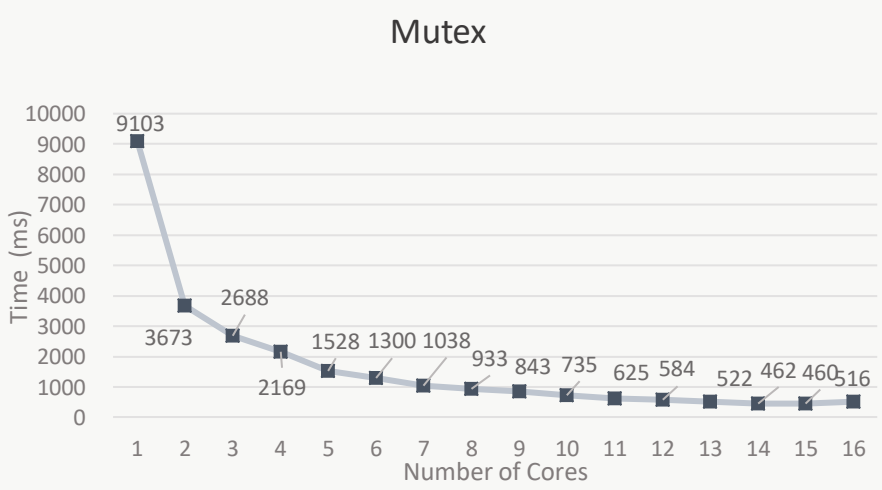


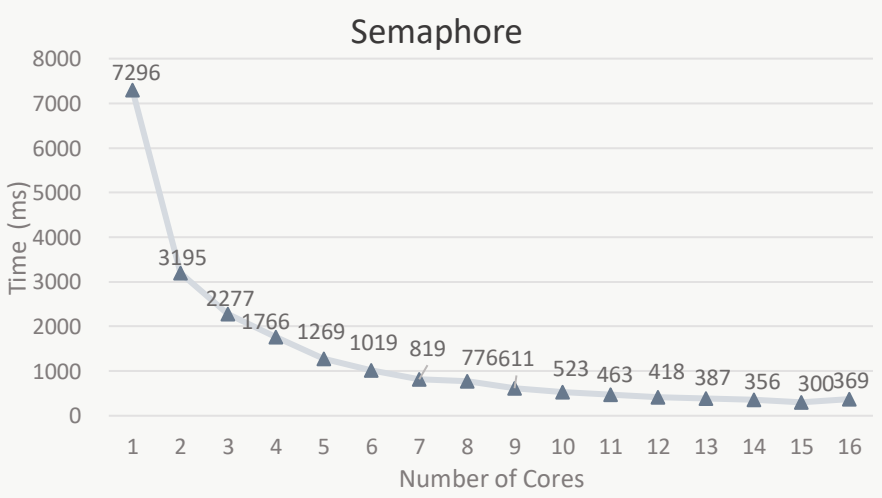
Increment Core Test

Execution Environment	
CPU	AMD Ryzen Threadripper 2950X
Cores	**Linear increment (1~16)**
Pairs	**256 pairs (512 processes or threads) [Fixed]**
Iterations	10,000
Unit	ms (milli second)



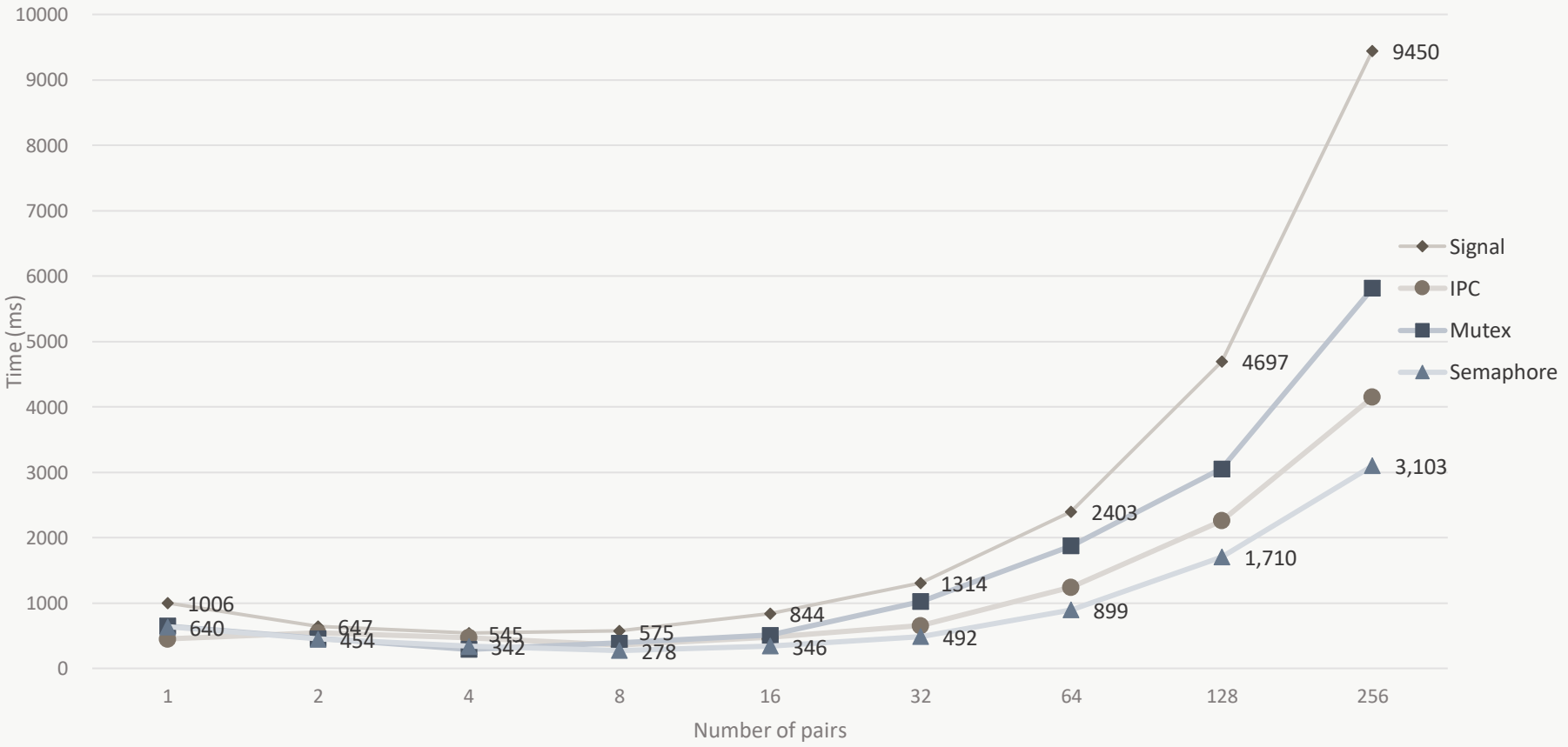






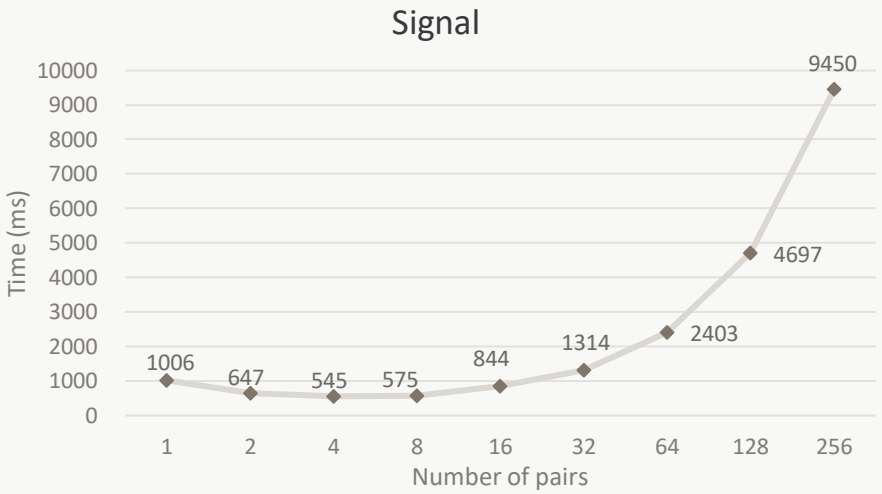
Increment Process Test

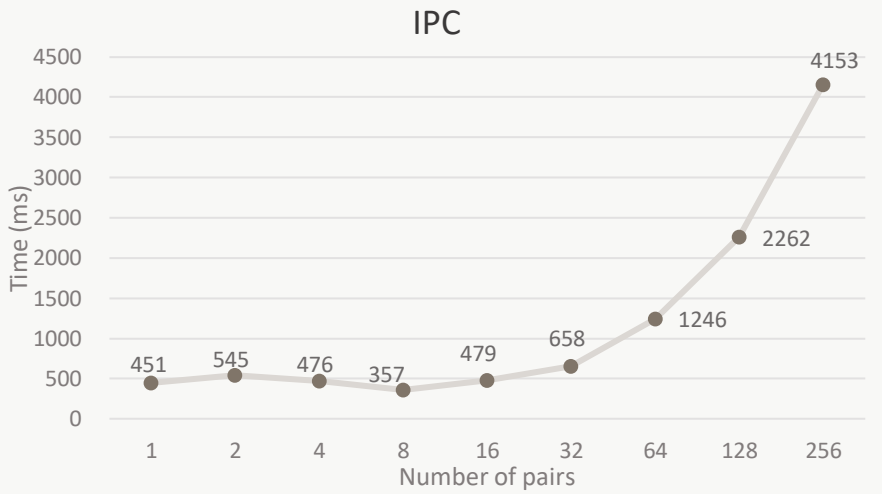
Execution Environment	
CPU	AMD Ryzen Threadripper 2950X
Cores	**16 Core [Fixed]**
Pairs	**Increment**
Iterations	100,000
Unit	ms (milli second)

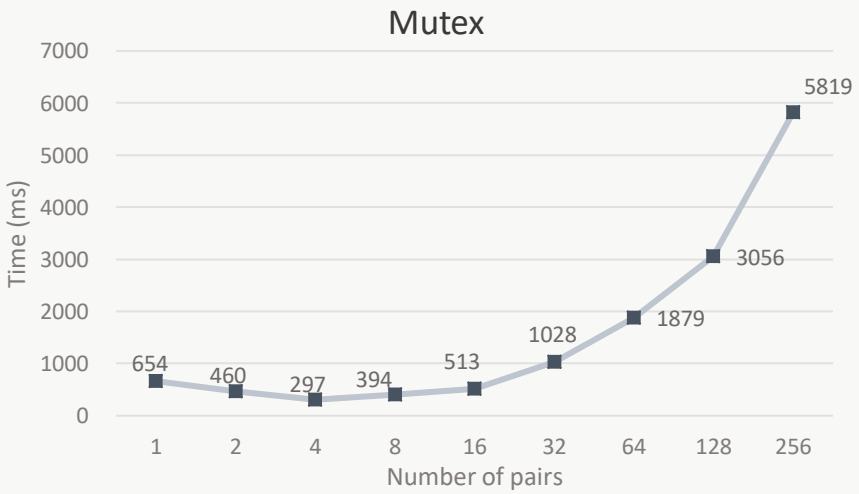


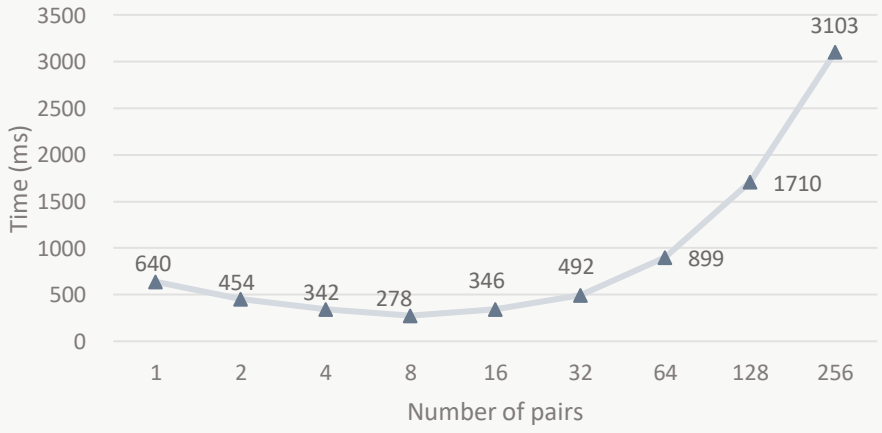
Increment Process Test

Execution Environment	
CPU	AMD Ryzen Threadripper 2950X
Cores	**16 Core [Fixed]**
Pairs	**Increment**
Iterations	100,000
Unit	ms (milli second)









측정 결과 분석

1. Increment Core Test

- 코어 수가 증가할 수록 Signal과 IPC의 실행 시간이 감소하는 것을 볼 수 있음.
- Semaphore는 15개 코어까지는 실행 시간이 감소하나, 16개의 코어에서 실행시간이 증가함을 볼 수 있었음.
- Mutex는 14개 코어까지는 실행 시간이 감소하나, 15개의 코어 이후에는 감소하지 않는 경향을 보임.

2. Increment Process Test

- 프로세스 수 증가에 따라서 프로세스 쌍의 개수가 1~8까지는 모든 시스템 콜이 감소하는 모습을 확인 할 수 있었음.
- 코어 수에 비해 프로세스 수가 충분하지 않으면 코어의 물리적 위치에 따른 성능 차이를 볼 수 있었음.
- 장기적으로 보았을 때는 Semaphore가 가장 성능이 좋음을 볼 수 있음.
- 가장 비효율적인 시스템 콜은 Signal로 보여짐.

추가 측정 진행

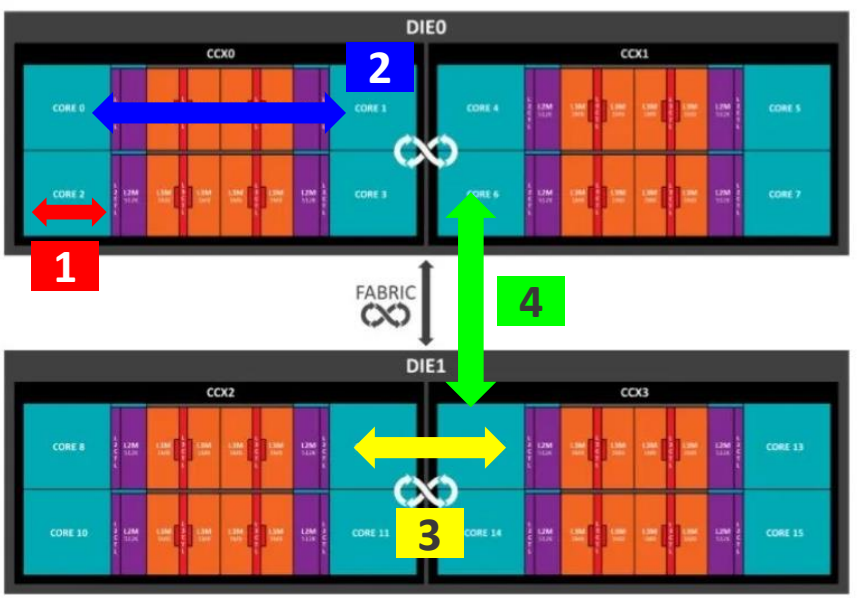
2. Increment Process Test

- **코어 수에 비해 프로세스 수가 충분하지 않으면 코어의 물리적 위치에 따른 성능 차이를 볼 수 있었음.**

3. Core Position Test

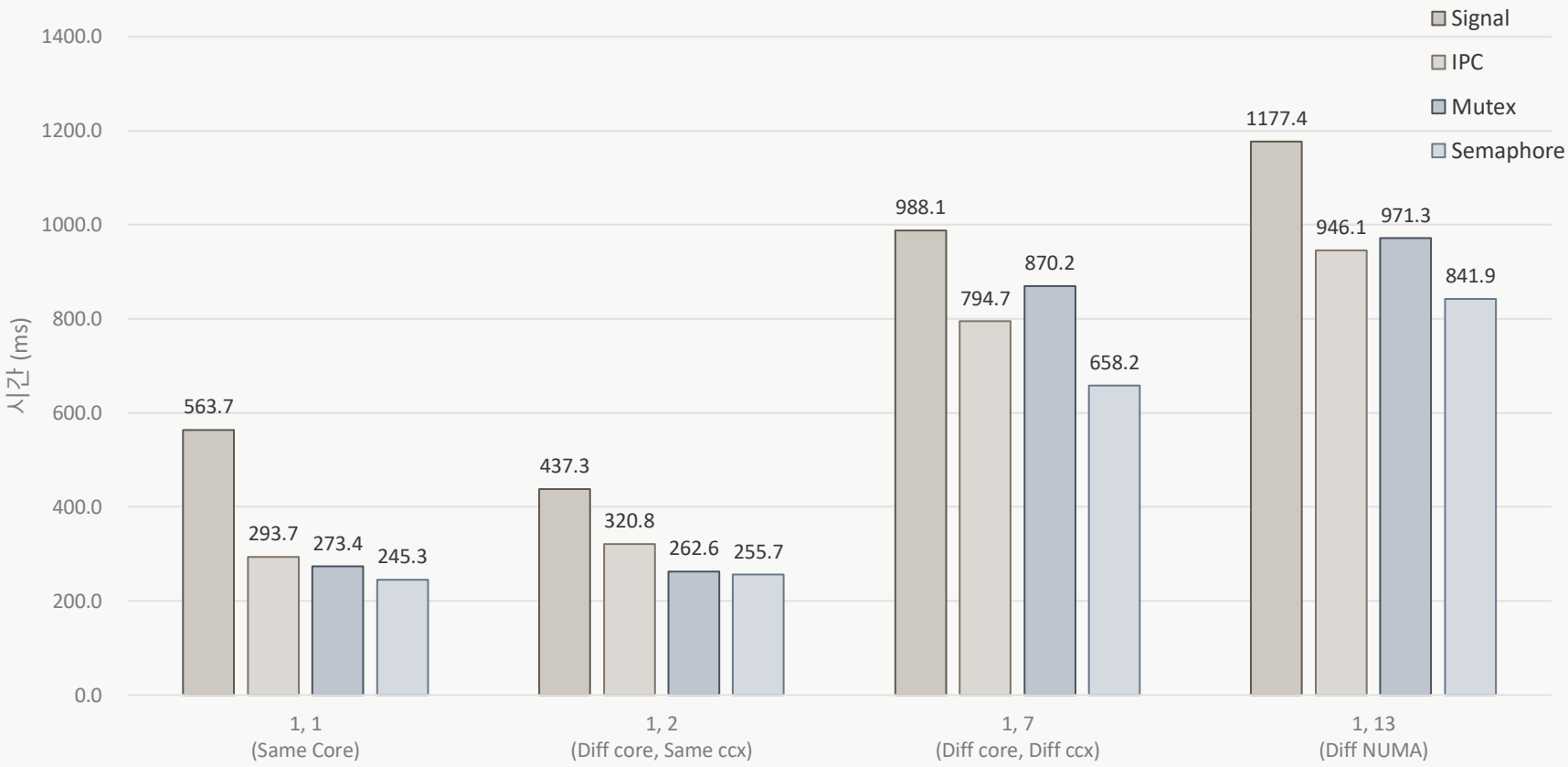
- 각 Ping하는 Process와 Pong하는 Process를 서로 다른 Core (Single Core 테스트 제외)에서 실행하게 만들어 코어 간의 위치에 따른 실행 시간을 측정

1. Single Core에서 실행
2. 같은 CCX의 다른 Core 에서 실행
3. 다른 CCX의 Core에서 실행
4. 다른 DIE의 Core에서 실행



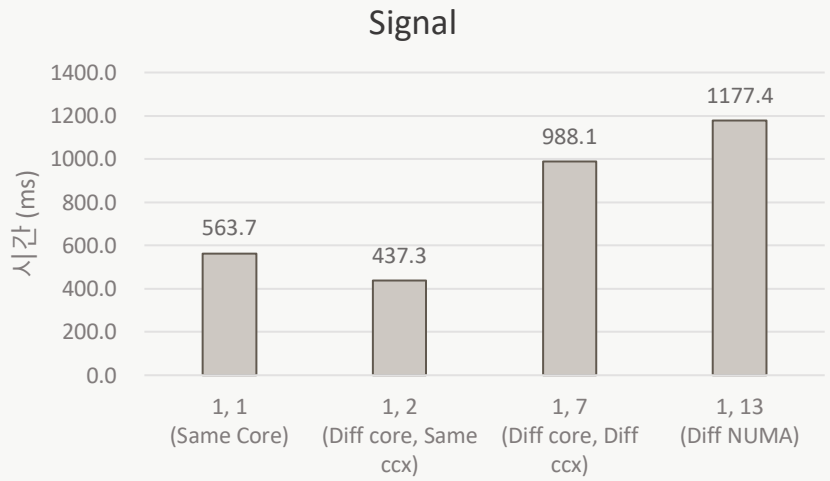
Core Position Test

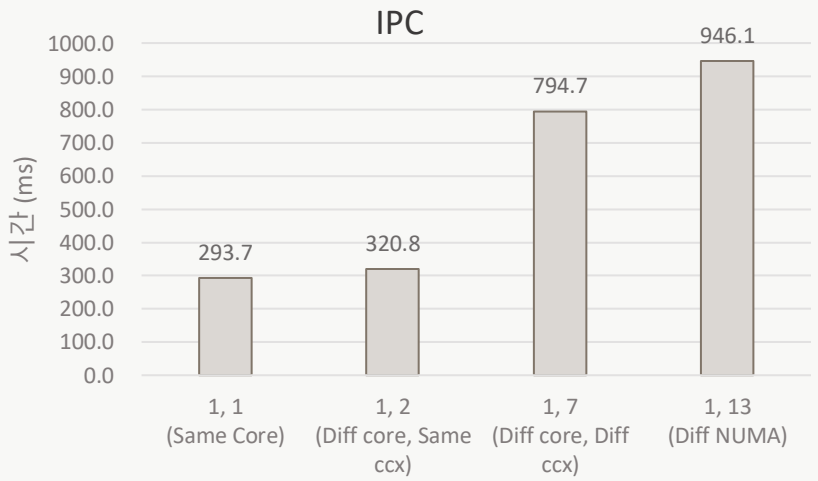
Execution Environment	
CPU	AMD Ryzen Threadripper 2950X
Cores	**2 Core [Fixed]**
Pairs	**1 pair (2 processes) [Fixed]**
Iterations	100,000
Unit	ms (milli second)

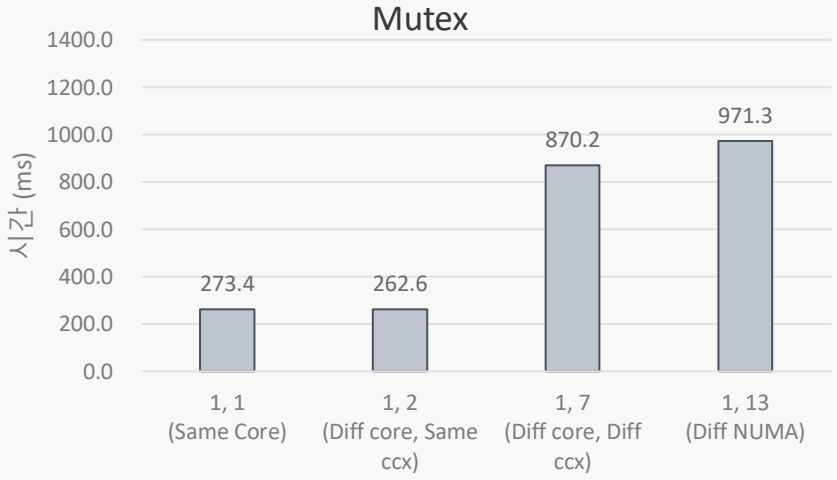


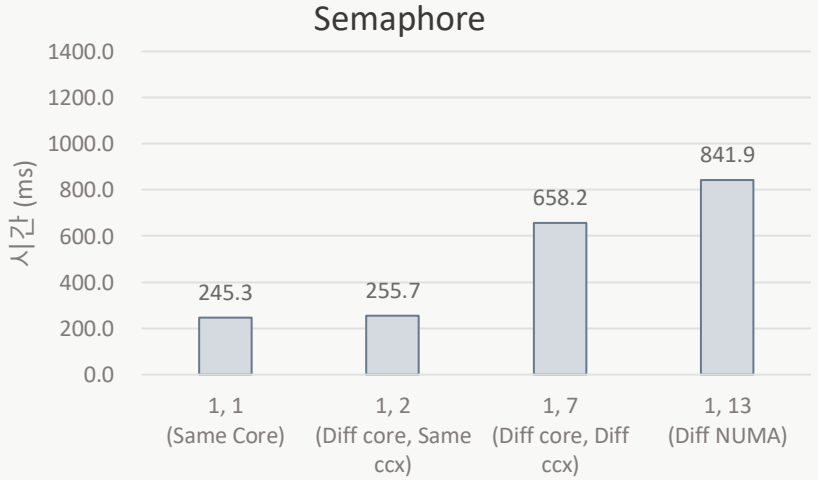
Core Position Test

Execution Environment	
CPU	AMD Ryzen Threadripper 2950X
Cores	**2 Core [Fixed]**
Pairs	**1 pair (2 processes) [Fixed]**
Iterations	100,000
Unit	ms (milli second)









추가 측정 결과

3. Core Position Test

- 같은 CCX 내에서 Single Core와 Dual Core의 결과
  - Dual-Core가 더 좋은 경우: Signal, Mutex
  - Single-Core가 더 좋은 경우: IPC, Semaphore

- 다른 CCX와 다른 DIE(NUMA)의 Core에서의 결과
  - 한 NUMA에 있는 경우가 다른 NUMA에 있는 경우보다 더 성능이 좋았음.

결론

- Semaphore나 Mutex의 경우는 15-core~16-core에서 오히려 성능이 하락하는 부분을 발견할 수 있었는데, 이에 따라 IPC와 Signal의 경우도 Core 개수를 16개 이상으로 증가시킬경우 성능이 더 좋아질 것이라 보장할 수 없습니다. 따라서 굳이 Core 수를 더 늘리더라도 성능 향상에는 한계가 있음을 알 수 있으므로, 무조건적인 하드웨어 증설을 권장하지 않습니다.

- Process/Thread 개수가 Core 수보다 적을 경우에는 Core 수와 같은 개수를 실행하는 것보다 성능이 오히려 더 안좋으므로, Core 수보다 더 많은 응용 Process/Thread를 실행/개발하는 것을 권장합니다.

- Multi-Core 환경에서 한 쌍이나 한 Topology로 구성되어 있는 Process/Thread들은 같은 Socket 내의 Core에 할당되어야 좋은 성능을 발휘할 수 있습니다.

Demo Scenario

1. 시스템 예외 처리 및 정상 작동 Scenario
   - Mode
     - Signal
   - Core 값 예외 처리
     - 0 (이하) –> 17 (초과) –> 3 (정상)
   - Process/Thread Pair 개수예외 처리
     - 0 (이하) –> 257 (초과) –> 12 (정상)
   - 패턴 반복 횟수 예외 처리
     - 0 (이하) –> 1000001 (초과) –> 100 (정상)
   - Benchmark 진행
     - Number of Test: 5
     - Graph: 1 (Per core graph)
   - Graph 출력
     - Y (Graph PNG 파일로 출력)

Demo Scenario

2. Graph 출력을 하는 Scenario
    - Mode
        - IPC
    - Core 개수
        - 8
    - Process/Thread Pair 개수
        - 8
    - 패턴 반복 횟수
        - 10000
    - Benchmark 진행
        - Number of Test: 2
        - Graph: 2 (Per process graph)
        - Graph Gap: 2
    - Graph 출력
        - Y (Graph PNG 파일로 출력)

Demo Scenario

3. Graph 출력을 하지 않는 Scenario
   - Mode
     - Lock - Semaphore
   - Core 개수
     - 10
   - Process/Thread Pair 개수
     - 1
   - 패턴 반복 횟수
     - 10000
   - Benchmark 진행
     - Number of Test: 2
     - Graph: 2 (Per process graph)
     - Graph Gap: 2
   - Graph 출력
     - N (Graph 출력 안함.)