KWANGWOON UNIVERSITY

2019년 2학기 **운영체제**

Assignment 2

System Software Laboratory
School of Computer and Information Engineering
Kwangwoon Univ.

Assignment 2-1

- **다음의 작업을 다중 프로세스/쓰레드로 수행하는 프로그램을 작성**
  - 프로그램 1. **numgen** (numgen.c)
    - Step 1. 특정 파일(“./temp.txt”)를 생성
    - Step 2. **i번째 값을 integer형 양수으로 생성할 프로세스 수의 2배만큼(MAX_PROCESSES)** 기록

```
...
#define MAX_PROCESSES 4
...
    FILE *f_write = fopen(...

    for(i=0; i<MAX_PROCESSES*2; i++)
    {
        fprintf(f_write, "%d", i+1);
        ...
    }

    fclose(f_write);
...
```

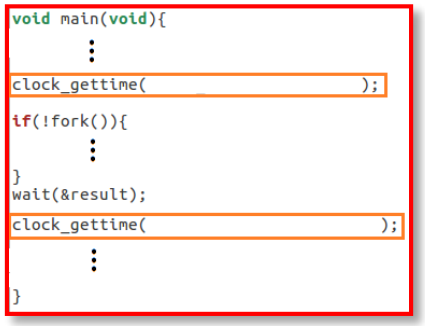
temp.txt

```
1
2
3
4
5
6
7
8
```

Assignment 2-1

- **다음의 작업을 다중 프로세스/쓰레드로 수행하는 프로그램을 작성**
  - 프로그램 2. fork.c, thread.c
    - Step 1. **MAX_PROCESSES** 만큼 프로세스 또는 스레드를 생성
      - **MAX_PROCESSES = 64**
    - Step 2. 최상단 프로세스/스레드마다 2개의 숫자를 읽음.
    - Step 3. 각 프로세스/스레드는 두 개의 숫자를 더한 후,
      부모 프로세스/스레드에게 값을 전달 (fork → **exit() 사용**)
    - Step 4. 최종적으로 나온 값, 전체 프로그램 수행시간 측정

- clock_gettime() 사용
- 결과에 대한 분석 내용 작성



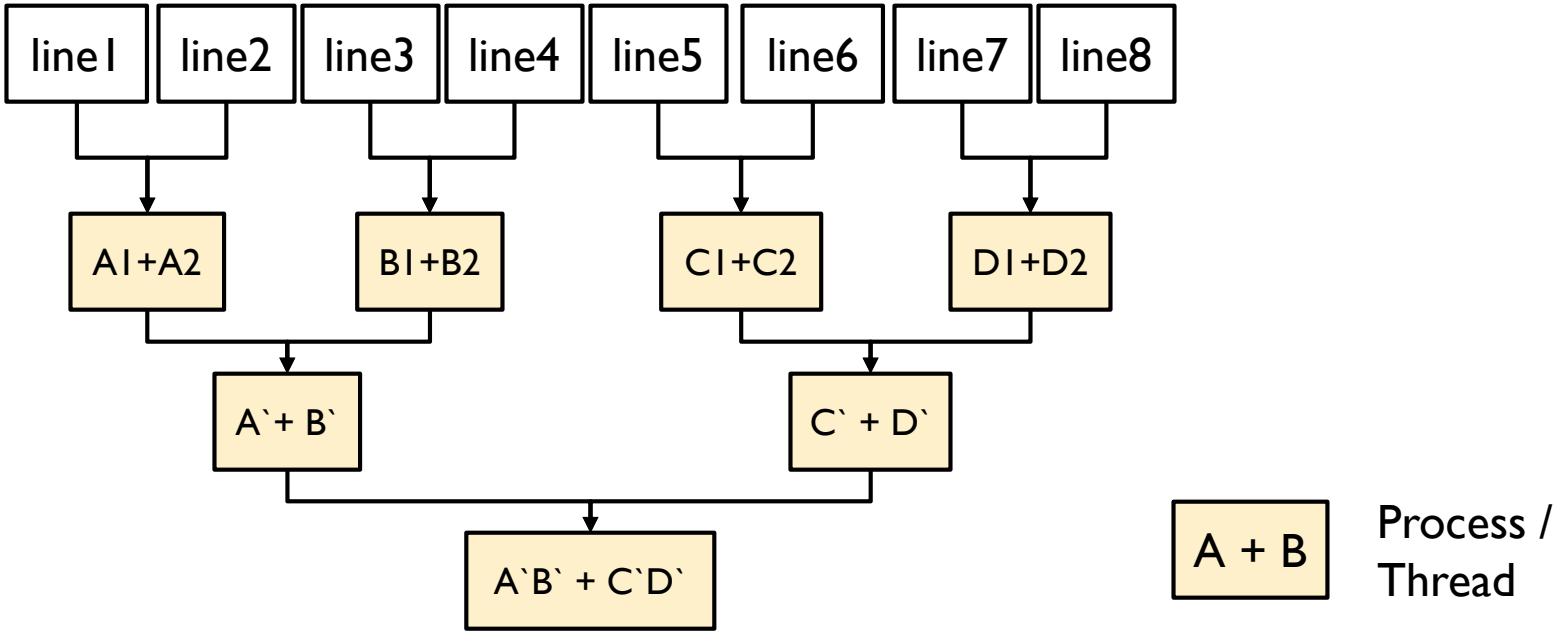
- 프로세스를 수행하는 파일(fork.c), 스레드로 수행하는 파일(thread.c) 각각 구현

Assignment 2-1

<Example>

- **numgen.c**
  - **Ex) MAX_PROCESS가 4이면, 숫자 8개를 기록**

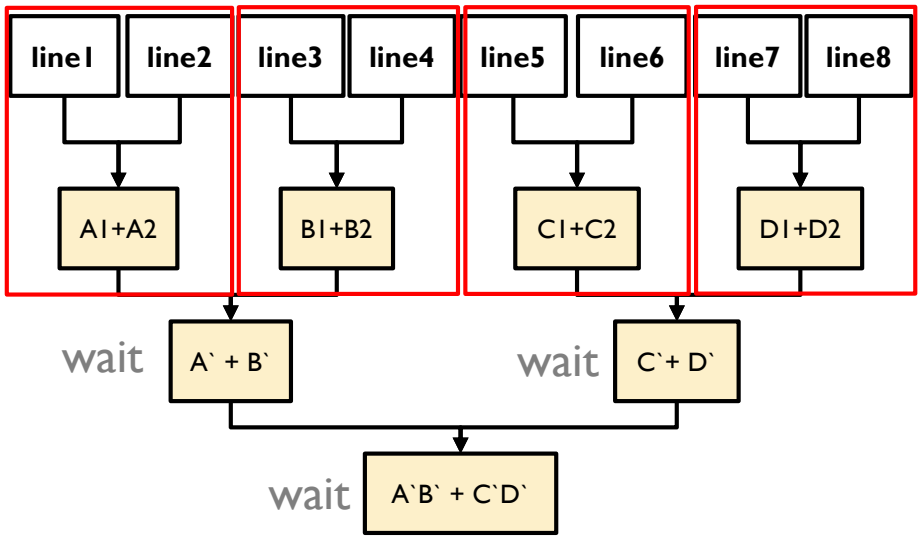
- **fork.c / thread.c**



Assignment 2-1

- **Requirements**
  - 메인 프로세스와는 별개로 각각의 연산을 수행하는 프로세스 or 스레드 생성
    - 하위 depth의 프로세스는 새로 생성하거나, 상위 depth의 프로세스 1개를 이용

- **Hint**



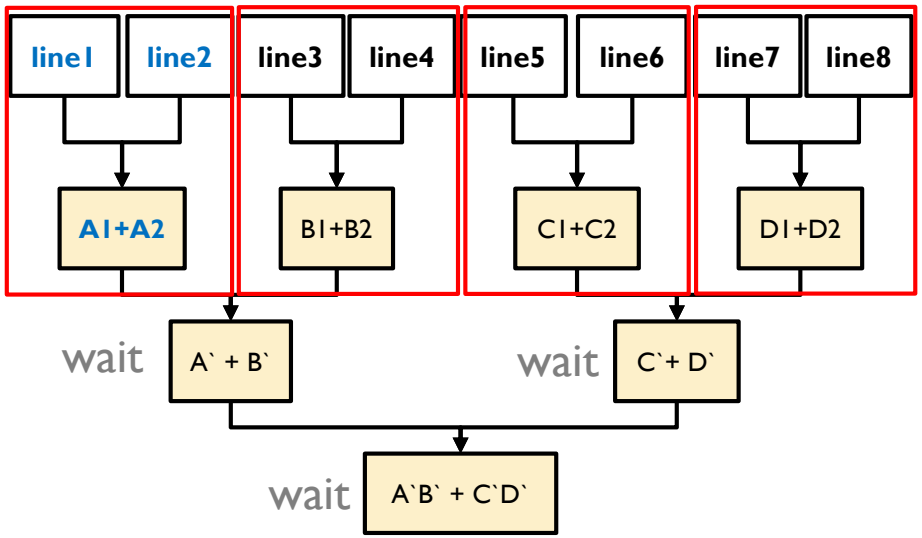
temp.txt

1
2
3
4
5
6
7
8

Assignment 2-1

- **Requirements**
  - 메인 프로세스와는 별개로 각각의 연산을 수행하는 프로세스 or 스레드 생성
    - 하위 depth의 프로세스는 새로 생성하거나, 상위 depth의 프로세스 1개를 이용

- **Hint**



temp.txt

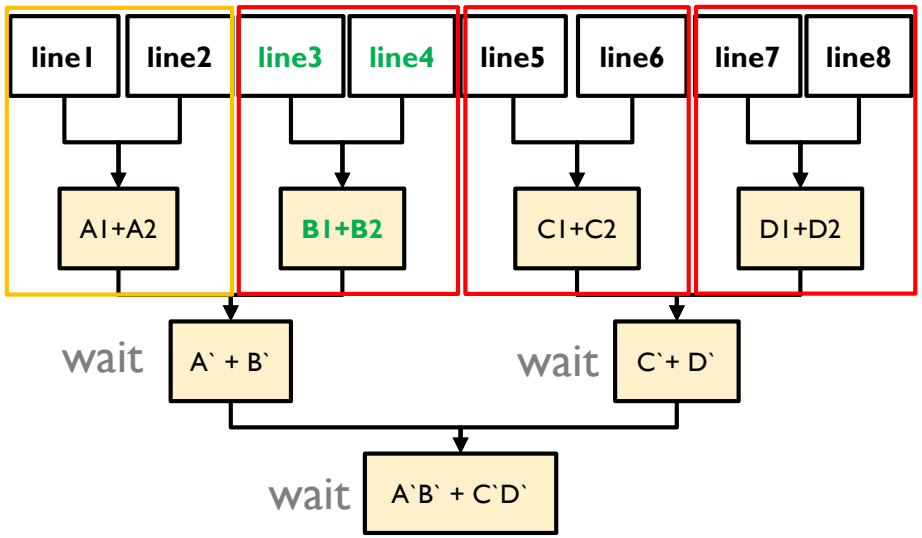
```
1
2
3
4
5
6
7
8
```

```
3
```

Assignment 2-1

- **Requirements**
  - 메인 프로세스와는 별개로 각각의 연산을 수행하는 프로세스 or 스레드 생성
    - 하위 depth의 프로세스는 새로 생성하거나, 상위 depth의 프로세스 1개를 이용

- **Hint**



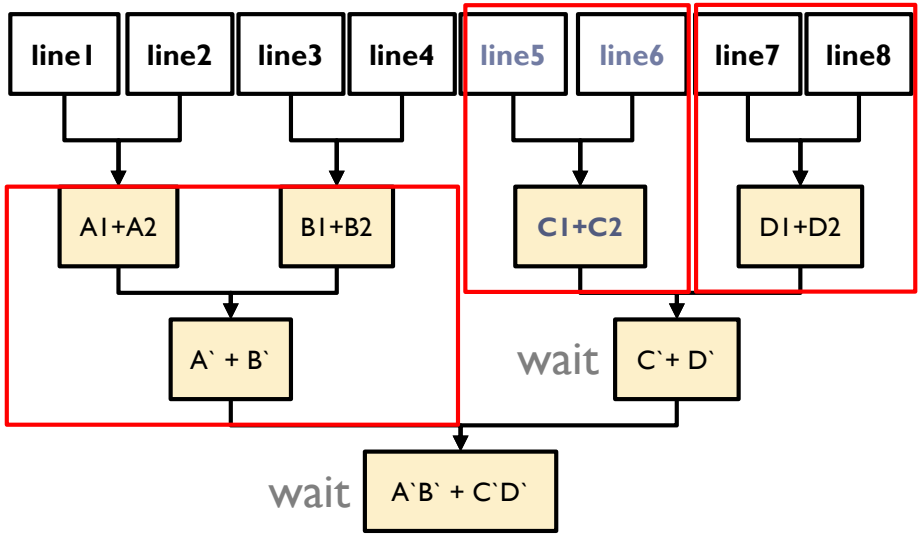
temp.txt

1 2 3 4 5 6 7 8
3 7

Assignment 2-1

- **Requirements**
  - 메인 프로세스와는 별개로 각각의 연산을 수행하는 프로세스 or 스레드 생성
    - 하위 depth의 프로세스는 새로 생성하거나, 상위 depth의 프로세스 1개를 이용

- **Hint**



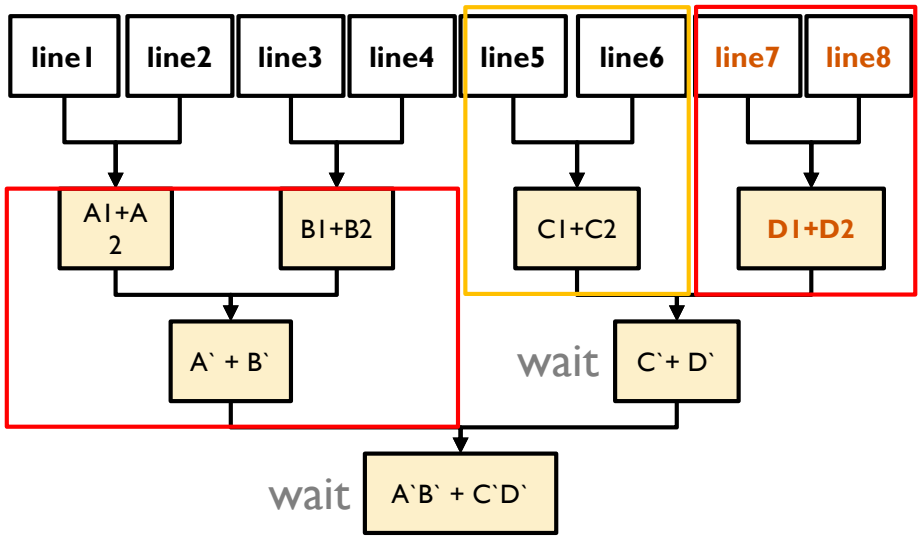
temp.txt

1 2 3 4 5 6 7 8
3 7 11

Assignment 2-1

- **Requirements**
  - 메인 프로세스와는 별개로 각각의 연산을 수행하는 프로세스 or 스레드 생성
    - 하위 depth의 프로세스는 새로 생성하거나, 상위 depth의 프로세스 1개를 이용

- **Hint**



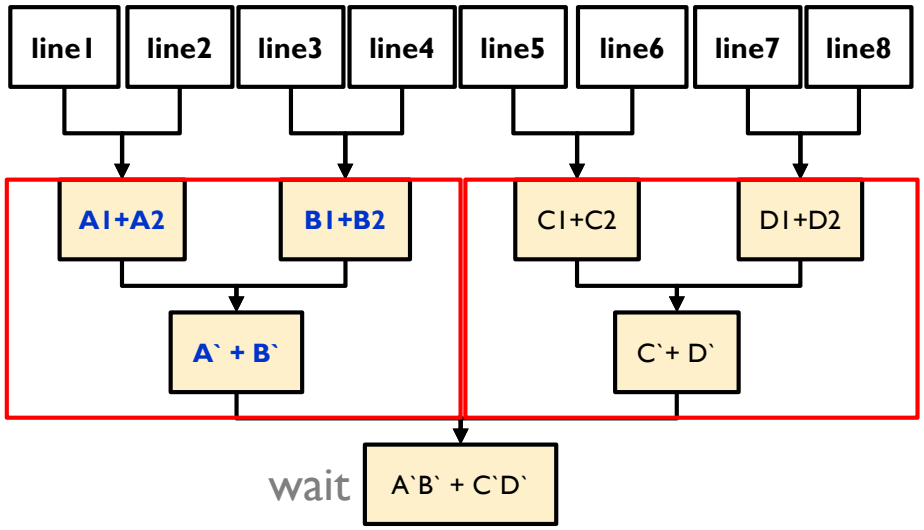
temp.txt

1 2 3 4 5 6 7 8
3 7 11 15

Assignment 2-1

- **Requirements**
  - 메인 프로세스와는 별개로 각각의 연산을 수행하는 프로세스 or 스레드 생성
    - 하위 depth의 프로세스는 새로 생성하거나, 상위 depth의 프로세스 1개를 이용

- **Hint**



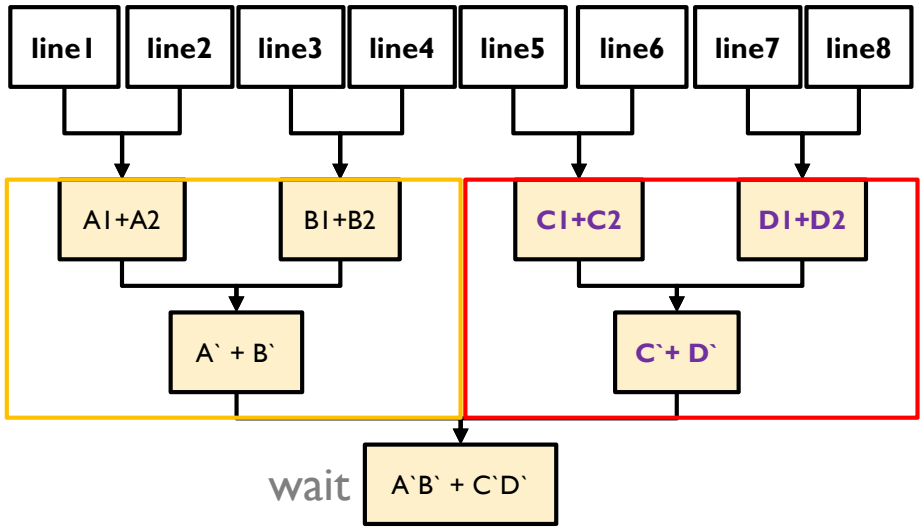
temp.txt

1 2 3 4 5 6 7 8
3 **7** 11 15 **10**

Assignment 2-1

- **Requirements**
  - 메인 프로세스와는 별개로 각각의 연산을 수행하는 프로세스 or 스레드 생성
    - 하위 depth의 프로세스는 새로 생성하거나, 상위 depth의 프로세스 1개를 이용

- **Hint**



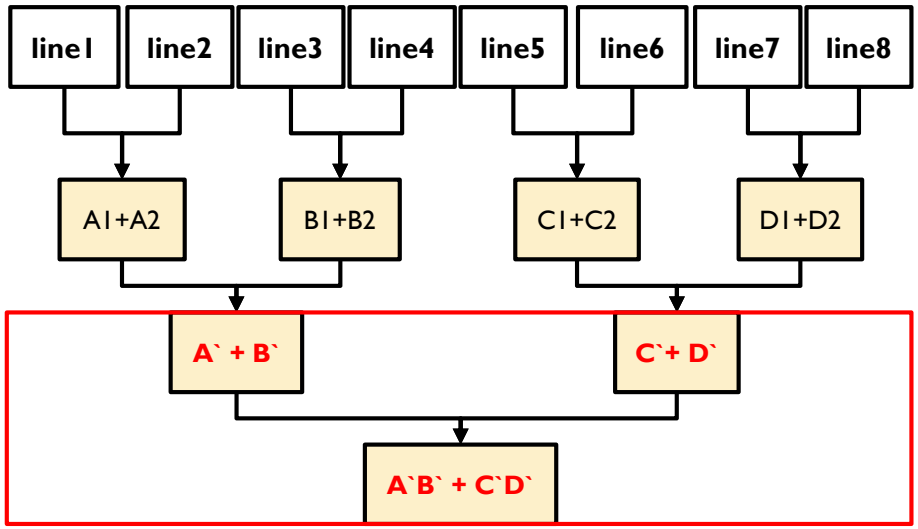
temp.txt

1 2 3 4 5 6 7 8
3 7 **11** **15** 10 **26**

Assignment 2-1

- **Requirements**
  - 메인 프로세스와는 별개로 각각의 연산을 수행하는 프로세스 or 스레드 생성
    - 하위 depth의 프로세스는 새로 생성하거나, 상위 depth의 프로세스 1개를 이용

- **Hint**



temp.txt

1 2 3 4 5 6 7 8
3 7 11 15 **10** **26** **36**

Assignment 2-1

- **다음의 작업을 다중 프로세스/쓰레드로 수행하는 프로그램을 작성**
  - Step 4. 최종적으로 나온 값, 전체 프로그램 수행시간 측정

```
os2019110613@ubuntu:~/test2$ ./a.out
value of fork : 136
0.002265
```

```
os2019110613@ubuntu:~/test2$ ./a.out
value of thread : 136
0.002423
```

< MAX_PROCESS = 8 >

```
os2019110613@ubuntu:~/test2$ ./a.out
value of fork : 64
0.016572
```

```
os2019110613@ubuntu:~/test2$ ./a.out
value of thread : 8256
0.013368
```

< MAX_PROCESS = 64 >

  - 결과에 대한 분석 내용 작성

Assignment 2-1

- **주의 사항**
  - 매 실험 전에 아래의 명령어를 수행할 것
    - 캐시 및 버퍼를 비워서 실험에 영향을 주는 요소를 제거

> - **rm –rf tmp***
> - $ **sync**
>   - Linux command to flush file system buffer
> - $ **echo 3 | sudo tee /proc/sys/vm/drop_caches**
>   - Linux commands to free pagecache, dentries, and inodes

  - 자식프로세스에서 부모프로세스로 값을 넘겨줄 때 (**exit ()**)

반환 값은 2^8 이상이면 안되며, 8-bit 만큼 right shift 해주어야 child process가 반환된 값을 정상적으로 확인할 수 있음. (e.g. a >> 8)

→ 이유를 보고서에 작성

Assignment 2-2

- **다음의 작업을 다중 프로세스/쓰레드로 수행하는 프로그램을 작성**
  - 프로그램 1. **filegen** (filegen.c)
    - Step 1. 특정 디렉토리(“./temp”)를 생성
    - Step 2. 이에 **무작위의 integer형 양수(≤9)**가 기록되어 있는 파일 (./temp/0, ./temp/1, ./temp/2, ...)을 **생성할 프로세스만큼(MAX_PROCESSES)** 생성

```
...
#define MAX_PROCESSES 10000
...
   for(i=0; i<MAX_PROCESSES; i++)
   {
      FILE *f_write = fopen(...
      fprintf(f_write, "%d", 1+rand()%9);
      fclose(f_write);
      ...
   }
...
```

Assignment 2-2

- **각 프로세스에서 CPU 스케줄링 정책을 변경**
  - 프로그램 2. **schedtest** (schedtest.c)
    - Step 1. **MAX_PROCESSES** 만큼 프로세스를 생성 → **fork() 사용**
    - Step 2. 각 프로세스에서 CPU 스케줄링 정책을 변경 (Hint. sched_setscheduler())
    - Step 3. 각 i(단, 0 ≤ i < MAX_PROCESSES) 번째 프로세스에서 미리 만들어져 있는 i번째 파일(./temp/i)에서 integer 데이터 읽음
  - Hint. 2-1의 fork.c에서 파일을 읽는 부분만 수정하여 사용
  - **성능을 비교할 수 있을 정도의 프로세스를 생성하여 실험할 것**
    - **MAX_PROCESS = 10000**

- **매 실험 전에 소스코드에 위치한 디렉토리에서 아래의 명령어를 수행할 것**
  - 캐시 및 버퍼를 비워서 실험에 영향을 주는 요소를 제거하기 위함

  - $ **rm -rf tmp***
  - $ **sync**
    - Linux command to flush file system buffer
  - $ **echo 3 | sudo tee /proc/sys/vm/drop_caches**
    - Linux commands to free pagecache, dentries, and inodes

- **linux에서 지원하는 3가지 CPU 스케쥴링와 Priority, Nice의 값을 각각 highest, defalut, lowest로 설정 후 결과에 대한 분석 내용 작성.**

  * the standard round-robin time-sharing policy
  * a first-in, first-out policy
  * a round-robin policy

Requirements (1/2)

- **Report**
  - Introduction : 5줄 이하
    - **Background 제외**
  - Reference : 각 과제별
    - e.g. 친구 도움, 책, 인터넷 사이트 주소 등.
    - 강의자료만 이용한 경우 생략 가능
  - Conclusion
    - **분석 결과 포함**

Requirements

- **Source Code**
  - 2-1
    - numgen.c // 테스트용 파일 생성 코드
    - fork.c // fork를 이용하여 구현한 코드
    - thread.c // pthread를 이용하여 구현한 코드
    - Makefile // 세 application을 모두 컴파일 하도록 작성
  - 2-2
    - filegen.c // 테스트용 파일 생성 코드
    - schedtest.c // 성능 측정 코드
    - Makefile // 두 application을 모두 컴파일 하도록 작성

Requirements (2/2)

- **Softcopy upload**
  - 보고서 및 소스파일은 하나의 압축파일로 압축하여 제출 (tar.gz)
  - **보고서 및 압축파일명**은 과제번호_학번 으로 수정
    - e.g. **(보고서)** **(압축 파일)**
      - OS_2_2017202001.pdf OS_2_2017202001.tar.gz

보고서는 "PDF"로 변환하여 제출

  - '19. 11. 03(일) 23:59:59 까지, [U-캠퍼스] ➔ [온라인참여학습] ➔ [과제제출]
    - Delay 없음
    - 업로드 양식에 어긋날 경우 감점 처리

  - **Hardcopy**
    - 제출하지 않음

  - 업로드 양식에 어긋날 경우 감점 처리
  - **과제 마감 당일 (11/03 일) 오후 5시 이전까지 발송한 질문에 한하여 답변**
  - ▪ 남건욱 (심동규 교수님 연구실 / ngotic@kw.ac.kr)
  - ✓ 위의 메일로 온 질문에만 답변합니다.