KWANGWOON UNIVERSITY

2019년 1학기 **시스템프로그래밍실습** 14주차

# Log file and Semaphore

**System Software Laboratory**
College of Software and Convergence
Kwangwoon Univ.

# Assignment #4 – description

- **TCP pre-forked server (Assignment 4-1)**
  - Process pre-forking
  - Signal processing

- **Process pool management (Assignment 4-2)**
  - Shared memory, mutex, pthread
  - Process scheduling (Max & Min bound)

- **Mutual Exclusion (Assignment 4-3)**
  - Log file 작성

# Semaphore

- **IPC의 일종**
  - 프로세스간 데이터를 동기화 하고 보호하기 위해 사용
  - 프로세스간 데이터를 공유하게 될 경우에 발생되는 동시 접근에 대한 문제를 해결

- **동작**
  - 초기화 시, 1 이상의 value값을 설정
  - 프로세스가 임계 영역에 접근 시, 1을 감소
    - Semaphore 값이 0이 되면 다른 프로세스의 접근을 중지
  - Semaphore를 반납할 때, value값을 1을 증가
    - 대기중인 다른 프로세스는 다시 1을 감소시키고 공유된 자원에 접근

# Semaphore APIs

- **System V semaphore**
  - 전통적인 세마포어 인터페이스
  - System V 운영체제(Unix version 5)에서 처음으로 도입
  - e.g. semget(), semctl(), semop(), etc.

- **POSIX semaphore**
  - POSIX 규격을 따르는 semaphore 인터페이스
  - **본 실습 및 과제에서는 POSIX semaphore를 이용**

# Semaphore APIs (cont'd)

- **sem_open**

```
#include <fcntl.h>
#include <sys/stat.h>
#include <semaphore.h>

sem_t *sem_open(const char *name, int oflag, mode_t mode, unsigned int value);
```

  - **Description**
    - Initialize and open a named semaphore
  - **Parameter**
    - name: 세마포어 식별자(이름)
    - oflage
      - O_CREAT: 존재하는 semaphore가 있을 경우 권한 습득
      - O_EXCL: 존재하는 semaphore가 있을 경우 -1 리턴
    - mode: 접근권한 (ex: 0755, 0777, etc.)
    - value: 세마포어에 접근가능한 프로세스 수 (1일 경우 Binary Semaphore)
  - **Returns**
    - On success, returns the address of the new semaphore
    - On error, returns SEM_FAILED

# Semaphore APIs (cont'd)

- **sem_close**

```
#include <semaphore.h>

int sem_close(sem_t *sem);
```

  - **Description**
    - closes the named semaphore referred to by sem.
  - **Parameter**
    - sem: address of the semaphore
  - **Returns**
    - On success, returns 0
    - On error, returns -1

# Semaphore APIs (cont'd)

- **sem_wait**

```
#include <semaphore.h>

int sem_wait(sem_t *sem);
```

  - **Description**
    - decrements (locks) the semaphore pointed to by sem.
    - 세마포어 값이 0보다 큰 경우, 값을 1 감소시키고 return
    - 세마포어 값이 0인 경우, 값이 증가할 때까지 block
  - **Parameter**
    - sem: address of the semaphore
  - **Returns**
    - On success, returns 0
    - On error, returns -1 (the value of the semaphore is left unchanged)

# Semaphore APIs (cont'd)

- **sem_post**

```
#include <semaphore.h>

int sem_post(sem_t *sem);
```

  - **Description**
    - increments (unlocks) the semaphore pointed to by sem.
  - **Parameter**
    - sem: address of the semaphore
  - **Returns**
    - On success, returns 0
    - On error, returns -1 (the value of the semaphore is left unchanged)

# Semaphore APIs (cont'd)

- **sem_unlink**

```
#include <semaphore.h>

int sem_unlink(const char *name);
```

  - **Description**
    - removes the named semaphore referred to by name.
  - **Parameter**
    - name: 세마포어 식별자(이름)
  - **Returns**
    - On success, returns 0
    - On error, returns -1

# Semaphore & file – sema.c

```c
#include <stdio.h>
#include <semaphore.h>
#include <fcntl.h>
#include <sys/types.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/wait.h>
#include <unistd.h>

const char * countFile = "count.db"; // output file

int main() {
        int fd;
        int i, num = 100;
        pid_t pid;

        sem_t *mysem = sem_open("mysem", O_CREAT | O_EXCL, 0700, 2); // Can access two process
        sem_close(mysem);

        fd = open(countFile, O_CREAT | O_TRUNC | O_RDWR, 0666);
        write(fd, (void*)&num, sizeof(num));
        close(fd);

        printf(">> start: 100\n");

        pid = fork();
```

# Semaphore & file – sema.c (cont'd)

```
if (pid > 0) { // parent process : increase number.
        mysem = sem_open("mysem", O_RDWR);
        for (i = 0; i < 10; ++i) {
                sem_wait(mysem);
                fd = open(countFile, O_RDWR);
                lseek(fd, 0, SEEK_SET);
                read(fd, (void*)&num, sizeof(num));
                printf("parent: %d\n", ++num);
                lseek(fd, 0, SEEK_SET);
                write(fd, (void*)&num, sizeof(num));
                close(fd);

                sem_post(mysem);
                usleep(100);
        }
        sem_close(mysem);
}
```

# Semaphore & file – sema.c (cont'd)

```
        else if (!pid) { // child process : decrease number
                mysem = sem_open("mysem", O_RDWR);
                for (i = 0; i < 10; ++i) {
                        sem_wait(mysem);
                        fd = open(countFile, O_RDWR);
                        lseek(fd, 0, SEEK_SET);
                        read(fd, (void*)&num, sizeof(num));
                        printf("child: %d\n", --num);
                        lseek(fd, 0, SEEK_SET);
                        write(fd, (void*)&num, sizeof(num));
                        close(fd);

                        sem_post(mysem);
                        usleep(100);
                }
                sem_close(mysem);
                exit(0);
        }
        wait(NULL);
        fd = open(countFile, O_RDWR);
        lseek(fd, 0, SEEK_SET);
        read(fd, (void*)&num, sizeof(num));
        close(fd);
        sem_unlink("mysem");
        return 0;
}
```

# Semaphore & file – Results

- **semaphore 동시 접근을 2로 설정**
  - 같은 결과를 보장할 수 없음

```
>> start: 100
parent: 101
child: 100
parent: 101
child: 100
parent: 101
child: 100
parent: 101
child: 100
parent: 101
child: 100
parent: 101
child: 100
parent: 101
child: 100
parent: 101
child: 100
parent: 101
child: 100
parent: 102
child: 101
parent: 103
child: 102
```

Parent와 Child 동시에 101 읽는다.
child: 101 – 1
parent: 101 + 1

Parent와 Child 동시에 102 읽는다.
child: 102 - 1
parent: 102 + 1

# Semaphore & file – Results (cont'd)

- **semaphore 동시 접근을 1로 설정**
  - 항상 같은 결과

```
>> start: 100
parent: 101
child: 100
parent: 101
child: 100
parent: 101
child: 100
parent: 101
child: 100
parent: 101
child: 100
parent: 101
child: 100
parent: 101
child: 100
parent: 101
child: 100
parent: 101
child: 100
parent: 101
child: 100
parent: 101
child: 100
```

sema.c의 해당 line 을 수정

```
sem_open("mysem", O_CREAT | O_EXCL  0700, 1);
```

KWANGWOON UNIVERSITY

2019년 1학기 **시스템프로그래밍실습** 14주차

# Assignment 4-3

**System Software Laboratory**
College of Software and Convergence
Kwangwoon Univ.

# Assignment 4-3

- **Process 관리 및 client 접속 정보를 log 파일에 기록하는 프로그램 작성**
  - Assignment 4-2 내용을 유지 하면서 아래에 추가된 내용을 콘솔에 출력 (Execution Example)
  - ① Client가 요청한 경로 (URL부분) (e.g. */hello*)
  - ② 연결 지속 시간
    - 클라이언트의 연결이 disconnected 될 때 출력
    - 단위: 마이크로 초(us)
    - 클라이언트 연결 종료 시간 까지만 출력!
      - 즉, sleep(5); 로 소요된 시간은 제외
  - 로그 파일(*server_log.txt*)에 추가로 작성
    - Log 작성 시, child 프로세스마다 생성한 thread를 이용하여 기록
    - 즉, thread 구동 함수가 string을 parameter로 받아 이를 file에 기록하게끔 구현

# Assignment 4-3 (cont'd)

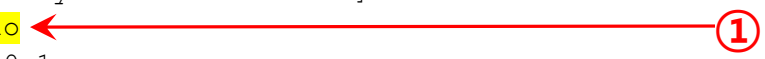
- **Process 관리 및 client 접속 정보를 log 파일에 기록하는 프로그램 작성** (cont'd)
  - 공유 자원인 로그 파일(*server_log.txt*) 동기화를 semaphore를 이용하여 보장
    - Semaphore를 사용하여 file에 접근 부분을 critical section으로 설정
    - 한 번에 하나의 thread만 접근 가능하도록 보장
    - Semaphore name은 할당된 학생별로 할당된 포트번호를 사용
  - Pre-forked server에 클라이언트가 연결되면, 접속 정보를 로그 파일(*server_log.txt*)에 작성
    - 예시 실행 결과를 참고

# Log file example

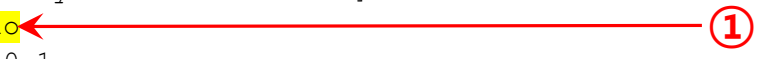
```
[[Fri May 31 12:58:49 2019] Server is started.
[Fri May 31 12:58:49 2019] 7786 process is forked.
[Fri May 31 12:58:49 2019] idleProcessCount : 1
[Fri May 31 12:58:49 2019] 7788 process is forked.
[Fri May 31 12:58:49 2019] idleProcessCount : 2
[Fri May 31 12:58:49 2019] 7790 process is forked.
[Fri May 31 12:58:49 2019] idleProcessCount : 3
[Fri May 31 12:58:49 2019] 7792 process is forked.
[Fri May 31 12:58:49 2019] idleProcessCount : 4
[Fri May 31 12:58:49 2019] 7794 process is forked.
[Fri May 31 12:58:49 2019] idleProcessCount : 5

================ New Client ================
TIME : [Fri May 31 12:58:56 2019]
URL : /hello
IP : 127.0.0.1
Port : 7786
PID : 16585
============================================

[Fri May 31 12:58:56 2019] idleProcessCount : 4

================ New Client ================
TIME : [Fri May 31 12:58:58 2019]
URL : /hello
IP : 127.0.0.1
Port : 7788
PID : 17097
============================================
```


# Log file example

```
[Fri May 31 12:58:58 2019] idleProcessCount : 3
[Fri May 31 12:58:58 2019] 8103 process is forked.
[Fri May 31 12:58:58 2019] idleProcessCount : 4
[Fri May 31 12:58:58 2019] 8105 process is forked.
[Fri May 31 12:58:58 2019] idleProcessCount : 5

============= Disconnected client ============
TIME : [Fri May 31 12:59:01 2019]
URL : /hello
IP : 127.0.0.1
Port : 16585
PID : 7786
CONNECTING TIME: 7589(us)
==============================================

[Fri May 31 12:59:01 2019] idleProcessCount : 6

============= Disconnected client ============
TIME : [Fri May 31 12:59:03 2019]
URL : /hello
IP : 127.0.0.1
Port : 17097
PID : 7788
CONNECTING TIME: 8124(us)
==============================================
```

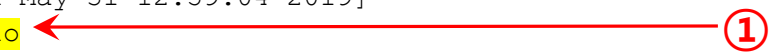







# Log file example

```
[Fri May 31 12:59:03 2019] idleProcessCount : 7
[Fri May 31 12:59:03 2019] 8105 process is terminated.
[Fri May 31 12:59:03 2019] idleProcessCount : 6
[Fri May 31 12:59:03 2019] 8103 process is terminated.
[Fri May 31 12:59:03 2019] idleProcessCount : 5

================= New Client =================
TIME : [Fri May 31 12:59:04 2019]
URL : /hello
IP : 127.0.0.1
Port : 7790
PID : 17609
==============================================

[Fri May 31 12:59:04 2019] idleProcessCount : 4
```

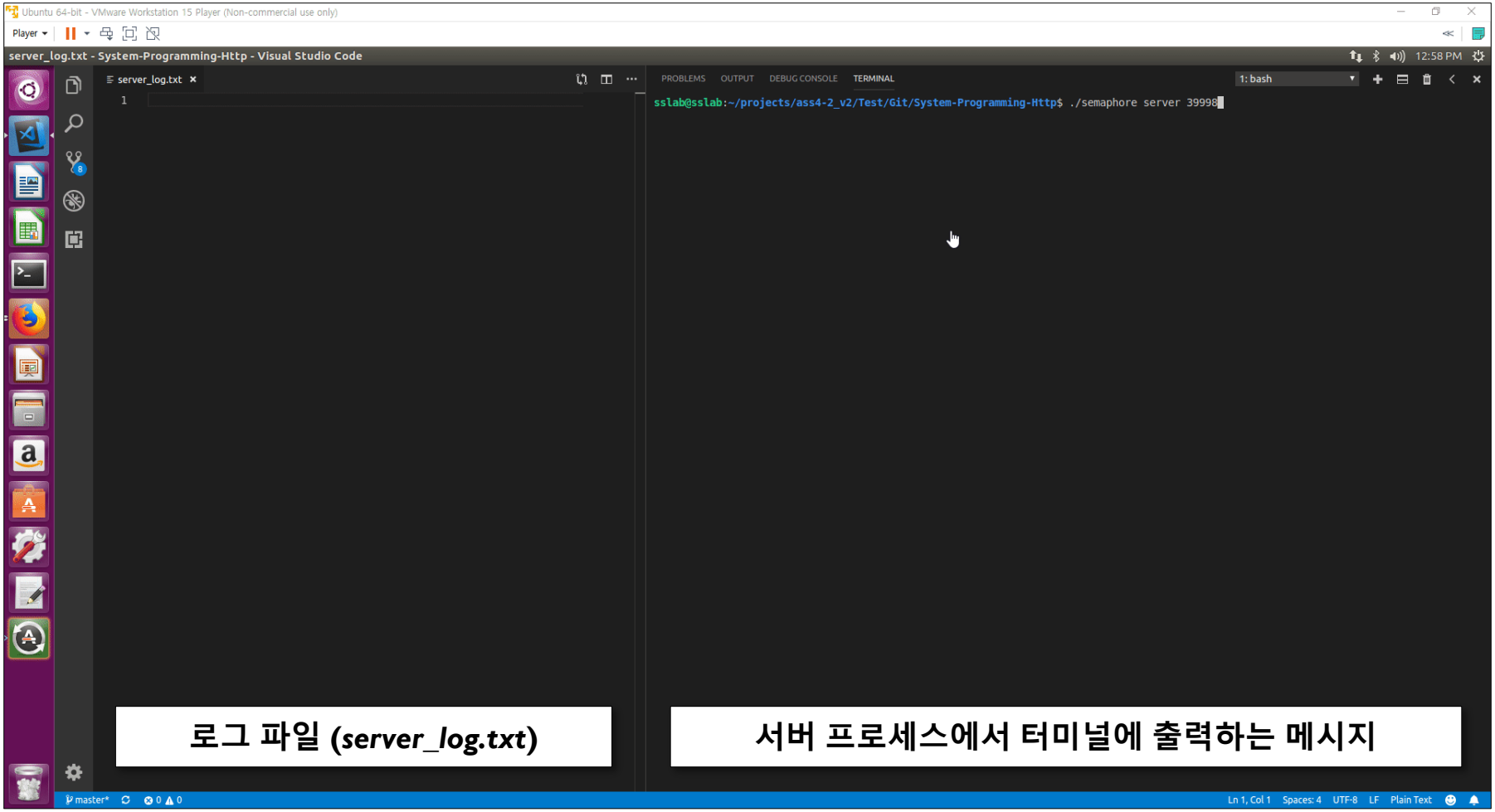


# Log file example

```
============= Disconnected client ============
TIME : [Fri May 31 12:59:09 2019]
URL : /hello
IP : 127.0.0.1
Port : 17609
PID : 7790
CONNECTING TIME: 187272(us)
=============================================

[Fri May 31 12:59:09 2019] idleProcessCount : 5
^c
[Fri May 31 12:59:15 2019] 7794 process is terminated.
[Fri May 31 12:59:15 2019] idleProcessCount : 4
[Fri May 31 12:59:15 2019] 7792 process is terminated.
[Fri May 31 12:59:15 2019] idleProcessCount : 3
[Fri May 31 12:59:15 2019] 7790 process is terminated.
[Fri May 31 12:59:15 2019] idleProcessCount : 2
[Fri May 31 12:59:15 2019] 7788 process is terminated.
[Fri May 31 12:59:15 2019] idleProcessCount : 1
[Fri May 31 12:59:15 2019] 7786 process is terminated.
[Fri May 31 12:59:15 2019] idleProcessCount : 0
```

① ② control + C 입력

# Execution example



# Assignment 4-3

- **Code requirements**
  - 이전 과제 부분에 문제가 있는 경우 감점
  - 출력 형식에 맞지 않으면 감점
  - 소스코드의 50% 이상 주석을 달지 않은 경우 감점

- **Makefile requirements**
  - 실행 파일이 "**semaphore_server_####**"로 생성되도록 Makefile 작성
    - ####은 본인의 포트 번호
  - 컴파일 도중 warning 발생 시 감점
  - "$ make" 를 통해 실행 파일이 생성되지 않는 경우, 0점

# Assignment 4-3 (cont'd)

- **Report requirements**
  - 보고서 표지: 실습 강의 시간, 담당 교수님, 학번, 이름 필히 명시
  - 아래의 내용을 보고서에 필히 포함
    - Introduction : 5줄 내외 작성
    - Flowchart : 본인이 작성한 코드에 대한 flowchart 작성
    - Pseudo code : 본인이 작성한 코드에 대한 pseudo code 작성
    - Reference
      - 참고한 자료 명시(웹 페이지, 친구 등)
      - 강의 자료만 활용한 경우 생략 가능
      - **Copy 발생 시 reference를 참고하여 채점**
    - Conclusion

- **과제 질문 관련**
  - 해당 과제 출제 담당 조교에게 이메일로 문의 ➔ **김태현 조교 (taehyun9203@gmail.com)**
  - 과제 제출 마감 당일에는 오후 4시까지 도착한 질문 메일에만 답변

# Assignment 4-3 (cont'd)

- **Softcopy upload**
  - 제출 파일
    - 보고서 (파일명: 실습요일_4-3_학번.**pdf**), ls.c, Makefile, [httpd.conf]
  - 위 파일들을 압축해서 제출 (파일명: 실습 요일_4-2_학번.tar.gz)
    - e.g. 월1,2 ➔ **mon**_4-3_2018722000.**tar.gz**
    - e.g. 화3,4 ➔ **thu**_4-3_2018722000.**tar.gz**
    - E.g. 금5,6 ➔ **fri**_4-3_2018722000.**tar.gz**
  - U-Campus의 과제 제출에 6월 14일(금) 23:59:59까지 제출
    - U-Campus에 올린 후 다시 다운로드 받아서 파일에 문제가 없는지 필히 확인
  - Delay 받지 않음
- **Hardcopy 제출하지 않음**

# 4차 퀴즈 안내

- **Date**
  - 2019/06/15(Sat)

- **시간 및 장소**
  - 오전 11:00 ~ 12:00 (단, 오전 11:30분 이후 입실 불가)
  - 장소
    - 새빛관 102호, 205호
    - 자세한 내용 공지사항 확인

- **내용**
  - 과제: 4차 과제
  - 강의 자료
    - 12, 13, 14주차13주차
    - 14주차