4190-307 운영체제 중간고사 이름
2015-04-29 학번

■ 다음 물음에 답하시오. 설명이 요구되는 질문에 대해서는 짧고 간략하게 답하시오. (각 5 점)

1. 운영체제가 어떤 계산환경에서 운용되느냐에 따라 중점 관리대상인 자원(resources)이 다르다. 일반 서버나 PC의 경우와 달리, 스마트폰의 경우에는 어떤 자원의 관리가 가장 중요한지 다음에서 하나만 고르시오. ① CPU ② 배터리 ③ 네트워크 ④ 그래픽 처리장치

2. 시분할시스템(timesharing system)은 다양한 계산기법을 이용하여 설계·구현된다. 다음에서 시분할시스템에 반드시 적용되어야 할 기법을 모두 고르시오. ① round robin scheduling ② graphical user interface ③ multiprogramming ④ 인터넷 ⑤ 병렬처리(parallel processing) ⑥ 다중 사용자(multi-user) 컴퓨팅

3. 현재 우리가 사용하고 있는 컴퓨터시스템은 대부분 미들웨어(middleware) 계층을 시스템 소프트웨어와 응용 소프트웨어 계층 사이에 두고 있다. 역사적으로 볼 때 미들웨어는 네트워크로 연결된 분산컴퓨터 시스템을 지원하기 위하여 생겨났다. 이러한 의미에서 미들웨어의 가장 기본적이고 중요한 기능은 무엇인가?

4. 컴퓨터시스템에서 각종 저장장치(storage devices)를 구성할 때, 계층적인 방법을 사용한다. 이렇게 저장장치를 계층화하는 이유 혹은 분류기준으로 크게 두 가지를 들 수 있다. 이 중 한 가지는 비용(혹은 가격)이다. 그러면 나머지 한 가지는 무엇인가?

5. 컴퓨터시스템에서 컴퓨터를 운용함에 있어 두 가지 모드를 설정한다. 즉 사용자(혹은 응용) 모드와 시스템(혹은 커널) 모드가 적용된다. 이렇게 두 개의 서로 다른 모드를 사용하는 이유를 모두 고르시오. ① 운영체제 혹은 커널을 보호하기 위하여 ② 시스템 성능을 향상시키기 위하여 ③ 각종 계산자원의 이용을 정확하게 혹은 효율적으로 하기 위하여 ④ 사용자 인터페이스의 반응 속도를 개선하기 위하여 ⑤ 실시간(real-time) 응용을 효율적으로 처리하기 위하여

6. 현대의 운영체제는 응용 프로그램의 개발자 입장에서 보면 플랫폼(platform)의 역할을 수행한다고 볼 수 있다. 이때 응용과 운영체제 사이를 이어주는 역할을 하는 기법(혹은 메카니즘, 구성요소)은 무엇인가?

7. 운영체제를 구성함에 있어 "적재가능한 모듈(loadable module)"을 이용할 수 있다. 이 접근방법의 장점과 가장 적합한 적용 부분(혹은 subsystem)은 무엇인가?

8. 시스템 콜을 구현할 때, 각 시스템 콜에 번호를 부여하여, 시스템 콜을 위한 테이블의 인덱스로 사용한다. 사용자 프로그램에서 시스템 콜을 호출할 때, 이 시스템 콜 번호는 커널에 어떻게, 즉 어떤 저장장치를 이용하여, 전달되는가? 또 시스템 콜의 인자(parameters/arguments)가 전달되는 방법을 두 가지만 드시오.

9. CPU에서 프로세스가 수행되다가 실행이 종료되지 않았는데도 불구하고 수행을 중단하는 사태가 발생할 수 있다. 그 이유로 적절하지 않은 항을 모두 고르시오. ① 입출력 요청 ② 타이머 인터럽트 ③ 자식 프로세스(child process)의 생성 ④ spin lock의 실행 ⑤ 새로운 프로세스의 도착 ⑥ time slice 혹은 time quantum의 종료

10. 프로세스 교체(switching)는 언제라도 일어날 수 있다. 그렇다면 (1) 어떤 기계어 명령이 수행중에 있을 때에도 가능한가? (예, 아니오) (2) 디스패처(dispatcher)가 하는 일 중, 가장 먼저 하는 일은 무엇인가?

11. 컴퓨터 시스템에서 가장 중요한 자료구조 중 하나가 프로세스 제어 블록(process control

block, PCB)이다. 다음 중에서 PCB의 항목으로 적절하지 않은 것을 하나 고르시오. ① program counter ② interrupt vector ③ CPU 사용시간 ④ child process ids ⑤ signal handler

12. 프로세스간 통신(IPC)을 메시지 전달(message passing) 기법에 의해 구현할 때, 만일 송신과 수신이 모두 blocking 모드인 경우, 이를 송신자와 수신자의 "랑데부(rendezvous)"라고 부른다. 랑데부의 경우 생산자-소비자 문제(producer-consumer problem)가 더 이상 이슈가 되지 않는다. 왜 그런지 설명하시오.

13. 부모 프로세스가 기다리지 않는데, 즉 wait()을 수행하지 않는데, 자식 프로세스가 종료하였다면, 그 결과 이 프로세스는 무엇이라 불리나? ① orphan ② zombie ③ init ④ process0

14. 웹서버나 인터넷 서버에서는 많은 사용자를 대상으로 한다. 이러한 계산환경에서 단지 프로세스에만 기반한 시스템에 비하여 쓰레드에 기반한 시스템의 장점은 무엇인가?

15. 다음에서 옳은 항을 모두 고르시오. ① 커널 레벨 쓰레드는 각각 별도로 스케줄할 수 있다. ② 쓰레드는 자신만의 스택 포인터를 가지고 있다. ③ 사용자 레벨 쓰레드는 입출력 요청(I/O requests)을 할 수 없다. ④ "thread pool"에서 주어지는 worker thread는 클라이언트가 서비스 요청을 하기 전에 미리 생성된다. ⑤ POSIX Pthreads는 오직 응용/사용자 레벨에서만 사용 가능하다.

16. 리눅스에서는 clone() system call을 이용하여 쓰레드처럼 동작하는 자식 프로세스를 생성한다. 그렇다면 이렇게 생성된 자식 프로세스가 어떻게 쓰레드의 특성을 갖게 만들 수 있나? 그 방법을 간략히 설명하시오.

17. 어떤 컴퓨팅 환경에서 새로운 작업(job)이 쉴새없이 시스템에 도착한다고 가정하자. 다음의 스케줄링 알고리즘에서 "starvation"의 가능성이 전혀 없는 것을 모두 고르시오. ① round robin ② shortest job first ③ preemptive scheduling ④ First Come First Served (FCFS)

18. 두 개의 태스크 A와 B가 리눅스 시스템에서 실행되고 있다고 하자. A와 B의 "nice" 값은 각각 -5와 +5이며, 또 A와 B는 각각 I/O-bound와 CPU-bound라고 한다. CFS scheduler에서 A와 B의 vruntime 값을 비교하시오. 즉 시간이 지남에 따라 어느 쪽이 더 작아지거나 커지거나 하겠는가? 가상의 실행시간(virtual runtime)인 vruntime은 태스크의 우선순위에 따른 decay factor와 연계되어 있다.

19. 실시간 스케줄링 알고리즘인 "rate-monotonic scheduling" 방식은 자주 수행되는 태스크에 더 높은 우선순위를 부여하여 스케줄링한다. 또 CPU 사용율(utilization rate)을 기준으로 하여 스케줄가능성 여부를 제시할 수 있다. 강의시간에 강조한 바와 같이, 이 알고리즘이 갖는 의의는 무엇인가? (답: 이론적으로 스케줄가능성(schedulability)를 증명할 수 있음)

20. 상호배제(mutual exclusion)를 직접 실현할 수 있는 방법이 아닌 것을 고르시오. ① 메모리에 저장된 해당 데이터를 공유데이터로 지정 ② mutex lock 사용 ③ 모니터 사용 ④ 스케줄러를 중단시킴 ⑤ 인터럽트 메커니즘을 중단시킴

21. 세마포어를 사용할 때, 프로세스는 임계영역(critical section)을 접근하기 전에 wait() 연산을 수행한다. 그리고 임계영역을 나오기 전에 signal() 연산을 수행한다. 이 두 개의 연산의 순서를 바꾸어, 먼저 signal()을 수행하고 후에 wait()를 수행한다고 하자. 이때 어떤 문제가 발생할 수 있을까? ① Starvation ② 복수의 프로세스가 동시에 임계영역에서 실행 가능함 ③ 상호배제가 보장됨 ④ Deadlock

22. 어떤 프로세스가 실행을 시작하여 종료될 때까지의 총 실행시간은 "blocking" 시간과 "preemption" 시간을 고려하여야 한다. 이 두 가지의 시간을 비교하시오.

23. 다음 중 올바른 항은 어느 것인지 모두 고르시오. ① 세마포어의 구현에 있어, 대기 큐(waiting queue)를 사용하면 busy waiting을 방지할 수 있다. ② counting semaphore는 절대로 binary semaphore로 사용될 수 없다. ③ binary semaphore는 상호배제를 구현할 수 있다. ④ counting semaphore는 한정된 수의 인스턴스를 갖는 자원에의 접근을 효과적으로 통제할 수 있다.

24. 두 개의 프로세스 P1과 P2가 다음과 같은 세마포어 연산을 수행한다고 하자.

```
P1:                    P2:
    S1;                    wait(s);
    signal (s);            S2;
```

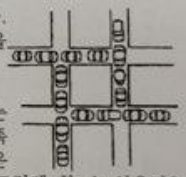
답:

여기에서 S1과 S2는 어떤 관계를 갖게 되는가?

25. 세 개의 프로세스에 12 개의 자원이 주어진다고 하자. 시간 0에 아래와 같은 상태에 놓여 있다. 이 테이블은 프로세스가 요구하는 최대 자원 수와 프로세스가 현재 소유하고 있는 자원의 수를 보이고 있다. 이 현 상태는 교착상태(deadlock)가 생기지 않고 "안전(safe)"하다고 할 수 있다. 그 이유를 설명하시오.

프로세스	최대 요구량	현재 소유량
P0	10	4
P1	3	2
P2	7	4

26. 교착상태(deadlock) 발생하기 위해서는 네 가지 조건이 만족되어야 한다. 이 중에서 wait-for 그래프를 이용하여 교착상태를 검출할 때 어느 조건을 검사하는가?



27. 그림에서 보이듯이 모든 교차점이 자동차가 꼬리를 물고 진입하여 막히는 현상을 "gridlock"이라고 부른다. 이 현상은 교착상태의 네가지 조건이 만족되었기 때문에 발생하였다. 이 문제를 해결하기 위한 가장 현실적인 방법은 네 가지 조건 중 어느 조건을 성립하지 않게 하는 것일까? 그리고 어떻게 그렇게 할 수 있을까? (답: hold and wait: 한 줄의 차를 후진시킴, circular wait도 맞는걸로 할 것)

28. 교착상태에 대처하기 위해 통상 상용 운영체제가 취하는 접근방법은?

■ 다음에 보인 프로그램에서 LINE C and LINE P 에서 출력되는 값은? (각 10점 - 총 20 점)

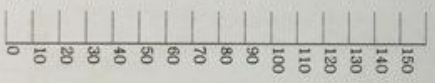
```
#include <pthread.h>
#include <stdio.h>
#include <types.h>

int value = 0;
void *runner(void *param); /* the thread */

int main(int argc, char *argv[])
{
        pid_t pid;
        pthread_t tid;
        pthread_attr_t attr;
```

```
        value = 10;
        pid = fork();
        if (pid == 0) { /* child process */
                pthread_attr_init(&attr);
                pthread_create(&tid,&attr,runner,NULL);
                pthread_join(tid,NULL);
                printf("CHILD: value = %d",value); /* LINE C */
                exit(0);
        }
        else if (pid > 0) { /* parent process */
                wait(NULL);
                printf("PARENT: value = %d",value); /* LINE P */
        }
}

void *runner(void *param) {
        value = 7;
        pthread_exit(0);
}
```

■ 다음의 테이블에 보인 프로세스와 그 특성에 대하여 (a) 알고리즘을 사용하여 프로세스들이 실행되는 상황을 아래의 차트에 그리시오. 우선순위 번호가 큰 것이 우선순위가 높은 것이다. (b) 평균 대기시간(average wait time)을 소수점 아래 한 자리까지 구하시오. (각 10점 - 총 20점)

Process	Arrival time	Burst	Priority
A	0.0000	50	6
B	20.0001	40	2
C	20.0001	20	9
D	40.0001	40	4

(a)



(b)

■ 다음의 물음에 답하시오. (각 10점 - 총 40점)

(a) 아래의 프로그램이 구현하는 것은 어떤 문제인가?

(b) 이 프로그램은 어떻게 상호배제 원칙을 실현하는가?

(c) 아래의 두 쓰레드는 각각 언제 종료되는가?

(d) 마지막 printf() 문의 출력은 무엇인가?

```
#include <sys/time.h>
#include <stdio.h>
#include <pthread.h>
#include <errno.h>

pthread_mutex_t produce_mutex = 0;  /* if 0, unlocked */
pthread_mutex_t consume_mutex = 0; /* if 1, locked */
int b;  /* buffer size = 1; */

main()
```

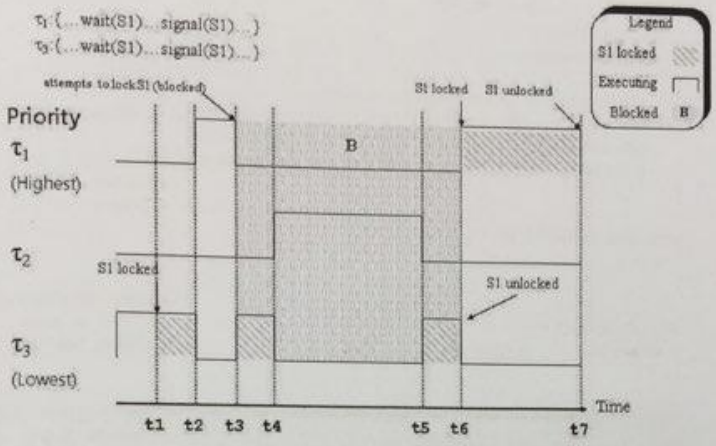
```
{
   pthread_t producer_thread;
   pthread_t consumer_thread;
   void *producer();
   void *consumer();

      pthread_mutex_lock(&consume_mutex);
      pthread_create(&consumer_thread,NULL,consumer,NULL);
      pthread_create(&producer_thread,NULL,producer,NULL);
      pthread_join(consumer_thread,NULL);   /* similar to wait() */
}

void add_buffer(int i){
   b = i;
}

int get_buffer(){
   return b ;
}

void *producer()
{
int i = 0;
printf("I'm a producer₩n");
while (1) {
   pthread_mutex_lock(&produce_mutex);
   add_buffer(i);
   pthread_mutex_unlock(&consume_mutex);
   i = i + 1;
}
pthread_exit(NULL);
}

void *consumer()
{
int i,v;
printf("I'm a consumer₩n");
for (i=0;i<100;i++) {
    pthread_mutex_lock(&consume_mutex);
    v = get_buffer();
    pthread_mutex_unlock(&produce_mutex);
    printf("got %d  ",v); // (d)
}
pthread_exit(NULL);
}
```

■ [우선순위 전도 문제(priority inversion problem)] (각 10점 - 총 30점)

(a) 시간 t2, t3, t4에서 일어나는 일들을 설명하시오.

(b) 우선순위 전도가 발생하는 시점은? 그 이유는?

(c) t5에서 τ1 대신에 τ3가 수행되는 이유는?



■ 다음의 프로그램은 bounded buffer problem을 모니터(monitor)를 이용하여 구현한다. (각 10점 - 총 30점)

(a) 이 예에서 상호배제가 어떻게 보장되는지 설명하시오.

(b) 프로그램 실행의 병행성(concurrency)이 어떻게 구현되는지 설명하시오.

(c) 아래의 프로그램에서 공란으로 남겨둔 부분을 알맞은 코드로 채우시오.

```
monitor bounded_buffer  {
        int items[MAX_ITEMS];
        int numItems = 0;
        condition full, empty;
        void produce(int v) {
                while (numItems == MAX_ITEMS) full.wait();
                items[numItems++] = v;
                empty.signal();

        int consume()  {
                int retVal;
                while (numItems == 0) empty.wait();
                retVal = items[--numItems];
                _____; /*(c) */
                return retVal;
        }
}
```

■ [보너스 문제] 수업/강의에서 가장 난해했던 내용은 무엇인가? 또 가장 흥미있었던 내용은 무엇인가? (5점)