운영체제 중간시험 풀이 (55 점 만점) 2017.4.21

1. 다음 빈 칸을 기술하라. (15 개 x 2 점 = 30 점)

(1) (다중 프로그래밍(multiprogramming)) 은 CPU 가 수행할 하나 이상의 여러 작업(코드와 데이터)을 메모리에 적재하여 빠른 작업의 전환이 가능하다.

(2) (가상화(virtualization))는 한 컴퓨터의 하드웨어가 다수의 다른 실행 환경(운영체제 등)을 제공하도록 추상화한다.

(3) 운영체제의 서비스 수행을 위해 직접 시스템 호출 대신하여 제공되는 대표적인 API 로 (POSIX API), Win32 API, Java API 등이 있다.

(4) (마이크로커널)은 모든 중요하지 않은 구성 요소를 커널에서 제거하고 시스템 및 사용자 수준 프로그램으로 운영체제를 구현하는 방식이다.

(5) 한 프로세스의 메모리는 텍스트, 데이터, (힙(heap)), 스택 영역으로 구분된다.

(6) (fork()) 시스템 호출을 하여 원래(부모) 프로세스의 주소 공간의 복사본으로 구성된 새로운 프로세스를 생성하고, (exec()) 시스템 호출로 생성된 프로세스의 메모리 공간을 새로운 프로그램으로 전환한다.

(7) 지명 파이프는 부모-자식 관계 없이 프로세스들 간에 접근이 가능하고, UNIX/LINUX 에서는 (FIFO) 라고도 한다.

(8) 다중 스레드(multithreading) 프로그래밍의 장점으로는 (응답성(responsiveness)), 자원의 공유(resource sharing), 경제성(economy), (규모의 가변성(scalability)) 등이 있다.

(9) 스케줄링에서 다음 CPU 버스트 길이 예측을 위해 사용되는 지수 평균(exponential average) $\tau_{n+1} = \alpha t_n + (1-\alpha)\tau_n$. 에서 t_n은 (직전에 실행된 CPU 버스트 길이) 이고 τ_n 은 (직전에 예측된 CPU 버스트 길이) 이다.

(10) (라운드로빈(Round-Robin)) 스케줄링 알고리즘은 각 프로세스에게 공평하게 동일한 CPU 사용 시간 조각(time slice)를 할당하여 스케줄링 하는 알고리즘으로 (시분할(대화형) 시스템)에 적합한 알고리즘이다.

(11) (처리기 친화성(processor affinity))은 다중 처리기 스케줄링 (multiprocessor scheduling) 시 캐시 사용의 효율을 높이기 위해 한 처리기에서 다른 처리기의 이주(migration) 를 피하고 대신 같은 처리기에서 프로세스를 실행을 시도하는 것이다.

2. 공유 메모리를 사용하는 유한 버퍼의 생산자-소비자 문제에서 공유 메모리에 아래와 같은 변수들이 저장되어 있을 때 다음을 기술하라. (10 점)

```
#define MAX_BUF 50
int buf[MAX_BUF];    // 생산자, 소비자가 주고 받는 원형 데이터 버퍼
int in = 0;          // 생산자가 데이터를 삽입할 버퍼 위치
int out = 0;         // 소비자가 데이터를 가져올 버퍼의 위치
```

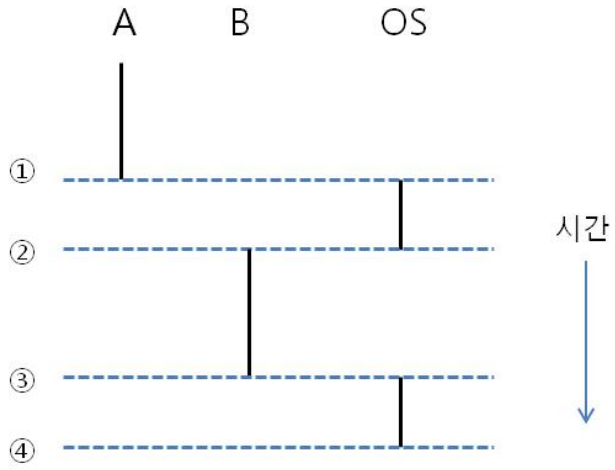
(1) 생산자 프로세스의 동작

```
int nextProduced;
while (true) {
    // .... 삽입할 데이터 생산
    while (((in+1) % MAX_BUF) == out)
          ;
    buf[in] = nextProduced;
    in = (in+1) % MAX_BUF;
}
```

(2) 소비자 프로세스의 동작

```
int nextConsumed;
while (true) {
    while (in == out)
        ;
    nextConsumed = buf[out];
    out = (out+1) % MAX_BUF;
    // .... 가져온 데이터 소비
}
```

3. 아래 그림은 사용자 프로세스 A, B 와 운영체제의 시간에 따른 수행을 보여 주는 그림이다. 초기 사태로 프로세스 A 가 실행 중이고, 시간 ① 에서 A 가 입출력 요구가 시작되고, ③에서 A 의 입출력 동작이 종료되었다고 가정할 때 다음을 기술하라. (5 개 x 3 점 = 15 점)



(1) ①과 ② 사이의 운영체제 동작

문맥 교환(context switch)으로 프로세스 A의 문맥을 프로세스 제어 블럭(PCB)에 저장하고, 운영체제가 스케줄링 알고리즘과 디스패치(dispatch)를 수행하여 프로세스 B의 PCB 내용을 복구하고, 프로세스 B에게 점프하여 제어권을 넘긴다.

(2) ② 에서 프로세스 A, B의 상태와 동작

프로세스 A는 대기 (waiting) 상태로 진입하여 입출력 동작이 종료하기를 기다린다.
프로세스 B는 실행 (running) 상태로 전환되어 CPU에서 실행 중이 된다.

(3) ③과 ④ 사이의 운영체제 동작

프로세스 A의 입출력 동작이 완료되어 장치 제어기로부터 인터럽트 신호가 CPU로 전달되어 운영체제는 장치와 메모리 간에 입출력 데이터를 전달하는 인터럽트 서비스 루틴을 수행한다.

(4) 스케줄러가 비선점형이라고 가정할 때 ④ 이 후의 동작

프로세스 A는 준비완료(ready) 상태로 전환하고, 프로세스 B의 실행이 종료되지 않은 경우 프로세스 B의 실행을 계속 한다. 만약 프로세스 B의 실행이 종료되거나 시간 조각(time slice)가 만료된 경우라면 스케줄러가 프로세스 A를 선택하여 실행 한다.

(5) 스케줄러가 선점형이라고 가정할 때 ④ 이 후의 동작

프로세스 A는 준비완료(ready) 상태로 전환하고, 프로세스 A의 우선순위가 높으면 프로세스 A가 B를 선점하여 실행 상태가 된다. 프로세스 B가 더 높은 우선순위이고 실행이 종료되지 않았다면 프로세스 B가 계속 실행 상태가 된다.