1.1 우리가 사용하는 교재 이름은 OSTEP (Operating Systems: Three Easy Pieces)인데, 이때 3개의 조각은 각각 ( ), ( ), ( ) 이다. Virtualization, Concurrency, Persistence

1.2 운영체제는 자원 관리자(resource manager)라고 정의되는데, 관리하는 자원에는 CPU, DRAM, Disk 같은 ( )과 태스크, 페이지, 파일 같은 ( )이 있다. Physical resources, Virtual resources

1.3 운영체제가 제공하는 인터페이스 또는 API를 ( ) 이라고 한다. System call

1.4 운영체제는 다양한 비 동기적인 사건들을 처리하는데, 이 중에서 하드웨어적인 사건을 ( )라고 하고 소프트웨어적인 사건을 ( ) 이라고 한다.

1.5 ( ) 이란 일의 량(amount of work)를 의미하며, 특히 컴퓨터에서는 작업들의 특성을 고려하여 얼마큼의 자원이 필요한지를 논할 때 사용하는 용어이다.

1.6 스케줄링 정책은 크게 두 가지로 구분할 수 있는데, ( )이란 중간에 작업을 멈추는 것 없이 끝까지 수행하는 정책이며, 반대로 ( )이란 중간에 수행을 멈출 수 있는 정책이다.

1.7 다중 프로세서 스케줄링에서 고려해야 할 것으로, ( )은 작업들을 여러 CPU들에게 균형 있게 분배하려는 것이며, ( )은 이전에 수행했던 CPU에서 다시 수행하려는 것이다.

1.8 주소 공간(address space) 중에서 쓰레드 간에 공유하지 않는 부분은 ( ) 이다. (다르 게 질문하면 쓰레드를 생성할 때마다 주소 공간 중 ( )은 새로 할당된다).

1.9 교착상태를 탐지할 때 생성하는 그래프 이름은 ( ) 이다. 여기서 노드(node)는 프로세스 또는 자원을 나타내고 에지(edge)는 할당 관계를 나타낸다.

1.10 ( )은 컴퓨터 시스템에서 자주 사용하는 최적화 방법으로, 비용이 발생하는 연산이 있으면 이것을 가능한 적게 수행하여 전체 부하를 줄이는 것이다.

1.11 (보너스) fork()와 exec()을 구분하는 것의 장점은? 확장성을 제공하기 위해서

2. 다음 질문에 답하세요. (총 12점)

2.1 3개의 프로세스를 생성하여 아래 프로그램을 수행시킬 때, 수행 결과를 쓰시오. 이때 쉘 에서 명령을 어떻게 입력하였는지 설명하시오 (4점, 단 각 프로세스의 pid를 11, 12, 13으로 가정, 동적으로 할당받은 메모리 주소는 0x10000으로 가정). /\* 문제 2.1번을 위한 프로그램 예 \*/ #include #include #include #include "common.h" int main(int argc, char \*argv[]) { int \*p = malloc(sizeof(int)); assert(p != NULL); printf("Address pointed to by %d: %p\n", getpid(), p); \*p = 1; while (1) { Spin(1); \*p = \*p + 1; printf("(%d) : %d\n", getpid(), \*p); } return 0; }/\* 문제 2.1번을 위한 프로그램 예 끝 \*/

2.2 2.1 문제에서 pid 11이 수행되다가 멈추고 pid 12이 시작되었을 때, 이 3개의 프로세스 들의 상태를 상태 전이도(state and transition)로 설명하시오 (3점, CPU는 1개만 있다고 가 정).

2.3 2.2 문제에서 Spin(1) 대신 sleep(1)으로 수정하면, 상태 전이도가 어떻게 달라지는지 논 하시오. 만일 새로운 응용들이 계속 생성되어 메모리 공간이 부족해지면 어떻게 되는지 논하 시오 (3점).

2.4 2.1 문제에서 만일 Spin(1)을 Spin(0.1)로 수정하면 (즉 1초동안 루프를 수행하는 것이 아니라 0.1초동안만 수행) 어떻게 되는지 논하시오 (2점).

3. 다음 질문에 답하세요. (총 10점)

3.1 제한된 직접 수행 (Limited Direct Execution)이 운영체제에서 중요한 이유 2가지를 설 명하시오. (2점). 이를 위해 운영체제가 활용하는 2가지 기법은 무엇입니까? (2점).

3.2 시분할 시스템(time sharing system)을 구현하기 위해 운영체제와 하드웨어는 다양한 방 법으로 협업합니다. 이러한 협업을 위해 부팅 과정과 프로그램 수행 과정에서 운영체제와 하 드웨어에서 어떤 작업들이 수행되는지 기술하시오 (4점).

3.3 컴퓨터 시스템은 일반적으로 하드웨어, 운영체제, 응용이라는 3개의 층으로 구성됩니다. 한편, 우리가 실습을 위하여 virtual box와 ubuntu를 설치하였는데, 이 경우에는 층 구조가 어떻게 되는지 그림으로 설명하시오 (2점).

4. 다음 질문에 답하세요. (총 18점) 4.1 다음과 같은 작업들이 존재한다고 가정할 때 FCFS, SJF, RR (time quantum은 1로 가 정), MLFQ (time quantum은 1로 가정)의 스케줄링 순서를 설명하시오 (4점). /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ 작업 도착 시간 서비스 시간 A 0 3 B 1 7 C 2 3 D 4 4 F 5 3 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

4.2 위 예에서 각 4가지 기법의 평균 응답 시간(response time)이 얼마입니까? (4점)

4.3 위 예에서 5.5 시간에 RR 기법에서 Ready queue에 존재하는 작업들을 순서대로 설명하 시오. MLFQ 기법에서 Ready queue에 존재하는 작업들을 순서대도 설명하시오 (5점).

4.4 다음과 같은 작업들이 존재한다고 가정할 때, Lottery와 Stride의 스케줄링 순서를 설명 하시오. (5점) /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ 작업 도착 시간 티켓 A 0 15 B 0 10 C 0 30 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ 5. 다음 질문에 답하세요. (총 20점)

5.1 아래 프로그램의 수행 결과를 설명하고, 이 프로그램을 사용하여 공유자원, 경쟁상태, 임 계영역, 원자성을 설명하시오 (6점). /\* 문제 5.1번을 위한 프로그램 예 \*/ #include "common\_threads.h" volatile int counter = 0; int loops; void \*worker(void \*arg) { int i; for (i = 0; i < loops; i++) { counter++; } return NULL; } int main(int argc, char \*argv[]) { if (argc != 2) { fprintf(stderr, "usage: threads \n"); exit(1); } loops = atoi(argv[1]); pthread\_t p1, p2; printf("Initial value : %d\n", counter); Pthread\_create(&p1, NULL, worker, NULL); Pthread\_create(&p2, NULL, worker, NULL); Pthread\_join(p1, NULL); Pthread\_join(p2, NULL); printf("Final value : %d\n", counter); return 0; } /\* 문제 5.1번을 위한 프로그램 예 끝 \*/

5.2 3개의 쓰레드 (T1, T2, T3)가 존재하며, 이들이 한 순간에 하나만 접근 가능한 공유 자원 을 동시에 접근하려 한다고 가정하자. 세마포어를 사용하여 병행성을 제어할 때, 1) 세마포어 의 초기값과 2) 세개의 쓰레드가 동시에 공유 자원을 접근하려고 할 때 세마포어 값의 변화와 3) 세개의 쓰레드가 수행을 완료하였을 때 세마포어 값의 변화를 설명하시오 (4점).

5.3 5.2 문제에서, 만일 한 순간에 하나의 쓰레드만 공유 자원에 접근 가능한 것이 아니라, 한 순간에 최대 2개의 쓰레드가 접근 가능하다면 (예를 들어 렌터카 회사에 차량이 2대임),

5.2의 대답이 어떻게 바뀌는지 논하시오 (4점). 5.4 Ticket lock의 동작 방식과 장점을 설명하시오 (3점). 5.5 MLFQ를 구현할 때 상호 배제의 필요성을 설명하고, coarse grained lock 기반 설계와 fine-grained lock 기반 설계 방식의 차이점을 MLFQ를 예로 들어 설명하시오 (3점). 6 다음 질문에 답하세요. (총 10점) 6.1 생산자-소비자 문제에서 고려해야 할 3가지 이슈는 무엇입니까? (2점) 6.2 아래 예제 프로그램이 정확하게 동작하는지, 만일 그렇지 않다면 어떻게 수정해야 하는지 논하시오 (3점). /\* 문제 6.2번을 위한 프로그램 예 \*/ int loops; cond\_t cond; mutex\_t mutex; void \*producer(void \*arg) { int i; for (i = 0; i < loops; i++) { Pthread\_mutex\_lock(&mutex); while (count == 1) Pthread\_cond\_wait(&cond, &mutex); put(i); Pthread\_cond\_signal(&cond); Pthread\_mutex\_unlock(&mutex); } } void \*consumer(void \*arg) { int i; for (i = 0; i < loops; i++) { Pthread\_mutex\_lock(&mutex); while (count == 0) Pthread\_cond\_wait(&cond, &mutex); int tmp = get(); Pthread\_cond\_signal(&cond); Pthread\_mutex\_unlock(&mutex); printf("%d\n", tmp); } } /\* 문제 6.2번을 위한 프로그램 예 끝 \*/ 6.3 기록자-판독자 문제와 생산자-소비자 문제의 차이점을 논하고, 이를 위하여 기록자-판독 자를 구현할 때 생산자-소비자 문제와 어떤 차이가 있는지 실제 코드로 예를 들어 설명하시오 (5점). 7. 다음 질문에 답하세요. (총 10점) 7.1 생각하는 철학자에서 교착상태가 발생할 수 있는데, 그 이유를 교착상태 발생의 4가지 조 건 (4 conditions)를 이용하여 설명하시오 (4점). 7.2 3개의 프로세스와 4개의 자원이 있는 시스템에서 안정 상태와 불안정 상태의 사례를 하 나씩 설명하고, 교착상태를 회피하기 위한 정책을 제안하시오 (3점). 7.3 pthread\_cond\_signal()에 비해 pthread\_cond\_broadcast()가 갖는 장점과 단점을 논하 시오 (3점)

1주차 1차시 퀴즈  
1. 운영체제(예를 들어 리눅스, 윈도우즈 등)와 응용 프로그램(예를 들어 워드, chrome 등)의 차이점을 "모드(mode)"라는 단어를 사용하여 설명하시오.   
2. 수업 교재인 OSTEP 1장에는 중국 철학자인 순자가 나옵니다. 순자가 한 말을 설명하시오.   
==> 응용 프로그램은 유저 모드에서 동작. 반면 운영체제는 커널 모드에서 동작한다.  
==> 듣는 것은 듣지 않은 것보다 좋고, 보는 것이 듣는 것보다는 좋고, 아는 것이 보는 것보다는 좋고, 해보는 것이 보는 것보다 좋다.    
  
1주차 2차시 퀴즈  
1. 운영체제는 "자원 관리자"라고 정의 됩니다. 운영체제가 CPU, DRAM, Disk라는 물리적 자원을 각각 어떤 논리적 자원으로 추상화 시켜 관리하는지 논하시오. (1점)  
2. 아래 프로그램 예에서 &의 기능을 설명하시오. (1점)    
==> CPU, DRAM, Disk를 각각 프로세스, 가상 메모리 (또는 페이지와 세그먼트), 파일 시스템 (또는 파일)로 추상화시켜 관리합니다.  
==> 백그라운드 수행 (background execution).   
한 명령어의 실행이 (다르게 표현하면 자식 프로세스가) 종료할 때까지 쉘이 기다리는 것이 아니라, 계속 진행하면서 다음 명령어 실행. 결국 여러 프로그램을 동시에 수행시킬 수 있음.     
  
2주차 1차시 퀴즈  
1. 가상 메모리와 물리 메모리의 차이점을, 오늘 동영상에서 배운 두개의 프로세스(pid는 각각 24113, 24114)를 사용하여 설명하시오. (1점)  
2. 디스크를 관리할 때 영속성을 고려해야 하는 이유를, 오늘 배운 Disk와 DRAM간에 차이점을 이용하여 설명하시오. (1점)  
==> 물리 메모리는 시스템에 하나만 존재. 반면 가상 메모리는 프로세스마다 존재 (이 예에서는 2개의 프로세스가 있으니 2개의 가상 메모리 존재)  
가상 메모리에서 사용하는 가상 주소는 주소 변환(address translation)을 통해 물리 메모리의 물리 주소로 바뀌어 접근함.   
따라서 두 개의 프로세스가 같은 가상 주소를 사용하더라도 서로 다른 물리 주소로 바뀌어 실제 물리 메모리에서는 다른 위치를 접근함.  
이 때문에 한 프로세스의 포인터 변경이 다른 프로세스에게는 보이지 않고 서로 독립적으로 동작함.    
==> Disk와 DRAM의 차이점 중에서 휘발성 여부 (volatile vs. non-volatile) 차이점이 영속성을 고려하게 만든다.  
(휘발성 여부 뿐만 아니라 속도, 용량, 인터페이스 등 다른 차이점도 영속성을 고려하게 만든다라고 답변하면 감점. 절반 점수만 받게 됩니다.)   
  
2주차 2차시 퀴즈  
1. PCB (Process Control Block)에서 관리되는 프로세스 정보를 설명하시오 (8가지 정보를 설명). (1점)  
2. 아래 프로그램이 수행될 때, 1) 10번째 라인에서 fork()에서 복귀한 후, 2) 15번째 라인 수행 중, 3) 18번째 라인 수행 중일 때, 부모와 자식의 상태 (state)를 각각 설명하시오 (부모가 자식보다 먼저 스케줄되어 수행된다고 가정). (2점)     
==> PCB에는 1) 프로세스 상태, 2) 식별자 (ID), 3) CPU 레지스터 정보, 4) 스케줄링 정보, 5) 메모리 정보, 6) 오픈된 파일 정보, 7) 입출력 상태 정보, 8) 계정 정보 등을 관리한다.   
==>   
- 10번째 라인에서 fork()에서 복귀한 후: 부모는 running, 자식은 ready (부모가 자식보다 먼저 스케줄되어 수행된다고 가정했기 때문)  
- 15번째 라인 수행 중: 부모는 waiting, 자식은 running  
- 18번째 라인 수행 중: 부모는 running, 자식은 terminated

3주차 1차시 퀴즈  
1. 트랩(trap)과 인터럽트(interrupt)의 차이점을 설명하시오. (1점)  
2. 아래 오른쪽 그림에서 컨텍스트 스위치가 몇번 발생하나요? 모드 스위치는 몇번 발생하나요? (1점)    
==>   
트랩과 인터럽트는 사건의 발생을 OS에게 알리는 메커니즘인데,   
트랩은 SW적인 사건 (0으로 나누기, 세그멘테이션 폴트, 시스템 호출 등)이며, 인터럽트는 HW적인 사건 (키보드, 장치, 타이머 등)을 의미한다.  
(또는 인터럽트는 trap와 HW 인터럽트로 구분되며, trap은 SW적인 사건이며, HW 인터럽트는 HW적인 사건을 의미한다라고 기술해도 정답으로 인정합니다.  
또는 트랩은 현재 수행중인 명령어와 관련된 SW적인 예외적인 사건이며, 인터럽트는 현재 수행중인 명령어와는 독립적인 HW적인 사건입니다도 정답) )    
==>   
컨텍스트 스위치는 4번 (X->Y, Y->Z, Z->X, X-> Y), 모드 스위치는 10번 (인터럽트 3번과 system call 2번, 각각 처리할 때 user->kernel 한번, kernel->user 한번으로 2번. 결국 모두 5\*2번)  
(참고로 user->kernel로 들어가는 것과 다시 kernel->user로 나오는 것을 한꺼번에 하나도 생각하여, 모드 스위치를 모두 5번이라고 답한 학생들이 많군요.   
정확하게는 각각을 한번씩으로 따로 카운팅 해 주어야 합니다. 음, 아쉬운 면이 있음을 고려하여 5번이라고 쓰면 0.2점만 감점하도록 하겠습니다.) 

3주차 2차시 퀴즈  
1. 선점형 스케줄링과 비선점형 스케줄링의 차이점을 설명하고, 운영체제 또는 실제 세상에서 각 스케줄링의 예를 제시하시오. (1점)   
2. 문맥 교환의 부하가 100ms일 때, 아래 그림 7.6과 7.7의 예에서 average turnaround time과 average response time을 계산하시오. (1점)   
==>   
선점형 스케줄링은 수행 도중 중단하였다가 재실행이 가능. 반면 비선점형 스케줄링은 한번 수행이 시작하면 끝날때까지 계속 수행됨.  
선점형 스케줄링의 예에는 RR, STCF, MLFQ 등이 있으며, 비선점형 스케줄링의 예에는 FIFO, SJF 등이 있음   
(실생활에 대한 다양한 응답들이 있었습니다. 예를 들어 농구장을 여러 팀들이 선점형으로 나누어 쓰는 것,  식당에서는 일단 식사가 시작되면 끝날때까지 테이블 사용을 보장하는 비선점형.)  
==>   
그림 7.6에서  
- ATT = (5 + 10 + 0.1 \* 1 + 15 + 0.1 \* 2)/3 = 10.1  
- ART = (0 + 5 + 0.1 \* 1 + 10 + 0.1 \* 2)/3 = 5.1  
그림 7.7에서  
- ATT = (13 + 0.1 \* 12 + 14 + 0.1\*13 + 15 + 0.1\*14)/3 = 15.3  
- ART = (0 + 1 + 0.1\*1 + 2 + 0.1\*2)/3 = 1.1  
  
  
4주차 1차시 퀴즈  
1. Interactive job과 Batch job의 차이점을 설명하고, 각각의 job에 적합한 스케줄링 정책의 이름을 쓰시오. (1점)  
2. 아래 왼쪽 그림의 시간 할당량이 1인 RR 정책에서, 8.5 시간에 준비 큐(ready queue)에 있는 프로세스들을 순서대로 설명하시오. (1점)  
==>   
Interactive job: 대화형 응용으로 응답 시간(response time)이 중요. RR, MLFQ 등과 같은 스케줄링 정책이 적합  
Batch job: 일괄 처리 응용으로 반환 시간 (turnaround time)이 중요. SJF, MLFQ 등과 같은 스케줄링 정책이 적합  
==>   
준비 큐에는 B->E->D가 존재, 참고로 C는 현재 수행 중인 상태라 ready queue에는 존재하지 않고 빠져 있습니다. A는 Zombie 상태    
  
  
4주차 2차시 퀴즈  
1. 다중프로세서 환경에서 스케줄링은 다음 두가지를 고려해야 한다.   
첫째는 기존에 수행했던 CPU에서 다시 그 작업을 수행시키려는 (   )이고, 둘째는 모든 CPU들에게 공평하게 작업을 분배하는 (  )이다. (1점)  
2. 아래 그림에서 각 가상머신(VM)의 지분이 2, 3, 5일 때 stride policy가 어떻게 스케줄링하는지 설명하시오. (1.5점)   
==>  
캐시 친화도(Cache affinity), 부하 균등 (load balancing)   
==>   
VM  A,  B,  C   
지분 2, 3,  5   
Stride 15, 10, 6 (최소 공배수인 30을 지분으로 나눔. 학생들의 응답에서는 다른 공배수 사용하여 다른 stride도 가능)  
  
 Scheduling order  
Pass value  0, 0,  0      ==> A  
 15, 0,  0      ==> B  
 15, 10, 0      ==> C  
 15, 10, 6   ==> C  
 15, 10, 12 ==> B  
 15, 20, 12 ==> C  
 15, 20, 18 ==> A  
 30, 20, 18 ==> C  
 30, 20, 24 ==> B  
 30, 30, 24 ==> C  
 30, 30, 30         ==> 지금까지 2:3:5보장. 이후 동점에서 다시 반복하기 때문에 위 순서와 동일한 순서 보장

5주차 1차시 퀴즈  
1. 쓰레드 모델에서 공유 자원(shared resource)과 독점적인 자원(exclusive resource)를 설명하고, pthread\_create()의 역할을 논하시오. (1점)  
2. 그림 26.6를 사용하여 1) 공유 자원, 2) 경쟁 상태, 3) 원자성, 4) 임계 영역, 5) 상호 배제를 설명하시오. (1점)    
==>   
공유 자원: code, data, heap, file vs. 독점적인 자원: 레지스터 (CPU 추상화), 스택  
pthread\_create()는 독점적인 자원만 새로 생성하고 공유 자원은 기존 쓰레드들과 공유한다.      
==>   
1) 공유 자원: counter 변수   
2) 경쟁 상태: p1, p2 쓰레드가 동시에 counter를 접근하려 함   
3) 원자성: counter 변수를 접근할 때 all or nothing 만족  
4) 임계 영역: 20번째 라인   
5) 상호 배제: 임계 영역 앞뒤에 lock, unlock 선언   
  
  
5주차 2차시 퀴즈  
1. Coarse-grained lock의 장점과 단점을 논하시오. (1점, fine-grained lock의 장단점도 논하시오)  
2. 그림 28.1의 락 구현 방법이 정확한지 그리고 공평한지 논하시오. (1.5점, 만일 정확하지 않거나 공평하지 않으면 그 이유를 설명하시오)   
==>  
Coarse-grained lock: 1) 장점: 락 개수가 적어 프로그램이 복잡하지 않다. 2) 단점: 경쟁 심화로 성능 저하의 가능성이 있다.  
Fine-grained lock: 1) 장점: 병렬성 향상으로 성능이 좋다. 2) 단점: 락 개수가 많아져 프로그램이 복잡해질 수 있고, 교착 상태 가능성이 커진다.   
==>   
정확하지 않다. test 부분과 set 부분이 구분되어 있어 (원자적이지 읺아) 이 사이에 다른 쓰레드가 간섭하면 상보 배제가 보장되지 않는다.  
공평하지도 않다. 한 쓰레드가 먼저 대기 중인데, 우선 순위가 높은 다른 쓰레드들이 도착한다면 먼저 대기 중이던 쓰레드가 계속 밀려 기아(starvation) 가능성이 발생한다.  
공평성을 보장하려면 ticket lock처럼 깨어나는 순서를 정해주어야 한다.    
  
  
6주차 1차시 퀴즈  
1. 그림 29.7(Concurrent Linked List)과 29.8(Concurrent Linked List: rewritten)의 중요한 2가지 차이점을 설명하시오. (1점)  
2. 키보드에서 입력을 기다리는 프로그램을 작성하고 있습니다. 기다릴 때 spin 또는 sleep lock 중에서 어떤 것을 사용하는 것이 좋은지 논하시오. (1점)  
==>   
1) 락의 범위: 29.7은 Coarse-grained lock이며 29.8은 Fine-grained lock의 예임  
2) 29.8은 lock과 unlock의 개수가 일치하여 버그의 가능성이 적다.   
==>  
키보드에서 키가 입력될 예상 시간에 따라 다르다. 만일 빨리 입력될 것이라면 spin lock이 유지하다.   
반면, 늦게 입력될 것이라면 sleep lock을 사용하여 CPU를 다른 쓰레드에게 주어 유용한 작업을 하는 것이 더 효과적이다.   
  
  
6주차 2차시 퀴즈  
1. 생산자 소비자 문제에서 고려해야 할 3가지 이슈를 논하시오. (1점)  
2. 아래 30.8 프로그램이 정확하게 동작하는지 여부를 논하시오. 만일 정확하지 않다면 그 이유를 설명하시오. (1점)   
==>   
1) 상호배제, empty case, full case  
2) 정확하지 않음: if 대신 while 사용하여 깨어났을 때 조건 만족 여부를 다시 확인해야 함, 두 개의 조건 변수 사용하여 깨어날 쓰래드를 명시할 수 있어야 함

7주차 1차시 퀴즈  
1. 그림 31.5에서 세마포어 값의 의미를 설명하시오. 그리고 이 값이 -2가 될 수 있는지 논하시오. (1점)  
2. 생산자 소비자 문제와 기록자 판독자 문제의 차이점을 논하시오. (1점)  
==>  
1) 1이면 사용가능, 0이면 사용 불가이며 대기자는 없음, -1이면 사용불가이며 대기자 1명  
-2도 가능함. 의미는 사용불가이며 대기자 2명임을 의미   
==>   
1) 병행성 여부 (판독자가 여러명 접근 가능), 2) 기록자의 기아 상태 고려 필요, 3) 구현할 때 락을 2개 사용 등...  
  
  
7주차 2차시 퀴즈  
1. 교착 상태에 대한 예방 전략에서 제시한 4가지 기법을 설명하시오. (1점)  
2. 오른쪽 아래의 자원 할당 그래프(resource allocation graph)에서 교착 상태가 발생하는지 여부를 논하시오. (1점)    
==>  
1) 점유 대기 예방 (예를 들어 차가 교차로 중간에서 서는 일 없앰),   
2) 비선점 예방 (교차로에서 막히면 뒤로 후진하여 교차로를 비워줌),   
3) 환형대기 예방 (오른쪽 차량 진입 우선),   
4) Lock free 방법으로 상호 배제 보장 (lock이 아닌 atomic 연산을 사용하여 상호배제 보장)   
==>   
발생하지 않음. P5 완료 가능. 완료 후 r4를 반납하면 P4가 할당받고 완료 가능. 그 이후 P2, P1, P3 순서로 모두 완료 가능하며 결국 교착 상태가 아님.