IT CookBook, 운영체제, 개정 3판 연습문제 해답

Chapter 1 컴퓨터 시스템의 소개_연습문제

1	①	**11**	④
2	④	**12**	④
3	④	**13**	①
4	①	**14**	④
5	④	**15**	③
6	①	**16**	①
7	④		
8	①		
9	②		
10	②		

Chapter 2 운영체제의 소개_연습문제

1	③	**11**	③	**21**	①	**31**	②
2	③	**12**	④	**22**	③	**32**	③
3	②	**13**	④	**23**	④		
4	①	**14**	③	**24**	③		
5	③	**15**	②	**25**	③		
6	③	**16**	①	**26**	③		
7	②	**17**	③	**27**	③		
8	④	**18**	②	**28**	④		
9	③	**19**	②	**29**	③		
10	①	**20**	③	**30**	②		

33. 운영체제의 주요 기능을 자원 관리와 시스템 관리로 나눠 기술하시오.

- 자원 관리 : 메모리 관리, 프로세스 관리, 주변장치(입출력장치) 관리, 파일(데이터) 관리
- 시스템 관리 : 시스템 보호(사용자 권한 부여), 네트워킹(통신), 명령 해석기 등 지원

34. 운영체제의 발전 목적은?

운영체제는 크게 편리성(사용자에게 편리한 환경 제공), 효율성(시스템 성능 향상), 제어 서비스 향상이라는 세 가지 목적에서 발전해 왔다.

35. 운영체제의 정의와 역할을 기술하시오.

- 정의 : 운영체제는 사용자가 응용 프로그램을 실행할 수 있는 기반 환경을 제공하여 컴퓨터를 편리하 게 사용할 수 있도록 도와주고, 하드웨어를 효율적으로 사용할 수 있도록 다양한 기능을 제공하는 소프트웨어이다.
- 역할 : 운영체제는 시스템 운영 요소를 적절하게 사용할 수 있도록 제어하면서(조정자), 각 응용 프로 그램에 필요한 자원(프로세스, 메모리, 파일, 장치 등)을 할당하거나 효율적으로 운영하려고 자 원을 할당하는 방법을 결정한다(자원 할당자 또는 관리자). 그리고 응용 프로그램과 입출력장치 를 제어한다(응용 프로그램과 입출력장치 제어자).

36. 프로세스를 관리하는 운영체제의 주요 활동은?

- 프로세스와 스레드를 스케줄링한다.
- 사용자 프로세스와 시스템 프로세스를 생성하고 제거한다.
- 프로세스를 중지하고 재수행한다.
- 프로세스 동기화 방법을 제공한다.
- 프로세스 통신 방법을 제공한다.
- 교착 상태deadlock를 방지하는 방법을 제공한다.

37. 파일을 관리하는 운영체제의 주요 활동은?

- 파일을 생성하고 삭제한다.
- 디렉터리를 생성하고 삭제한다.
- 보조기억장치에 있는 파일을 맵핑한다.
- 안전한(비휘발성) 저장장치에 파일을 저장한다.

38. 메모리를 관리하는 운영체제의 주요 활동은?

- 메모리의 어느 부분을 사용하고, 누가 사용하는지 점검한다.
- 메모리에 저장할 프로세스를 결정한다.
- 메모리를 할당하고 회수하는 방법을 결정한다.

39. 시분할 시스템 운영체제를 설명하고 장단점을 기술하시오.

시분할 시스템TSS, Time Sharing System은 다중 프로그래밍을 논리적으로 확장한 개념으로, 프로세서가 다중 작업을 교대로 수행한다. 다수의 사용자가 동시에 컴퓨터의 자원을 공유할 수 있는 기술이다.

[장점]

- 빠른 응답 제공
- 소프트웨어의 중복 회피 가능
- 프로세서 유휴시간 감소

[단점]

- 신뢰성 문제
- 보안 의문 및 사용자 프로그램과 데이터의 무결성
- 데이터 통신의 문제

40. 다중 프로그래밍 시스템과 다중 처리 시스템의 차이를 설명하시오.

- 다중 프로그래밍 시스템 : 프로세서(CPU) 하나가 둘 이상의 프로그램 처리
- 다중 처리 시스템 : 둘 이상의 프로세서(CPU)가 프로그램을 여러 개 처리

41. 다중 프로그래밍의 주요 장점은?

프로세서(CPU)가 수행할 작업을 항상 유지하여 프로세서 사용률(효율성) 증진

42. 운영체제의 사용자 서비스를 설명하시오.

① 사용자 인터페이스 제공 : 사용자와 컴퓨터 간의 상호작용이 발생하는 공간으로 CLI, 메뉴, GUI 등의 형태로 구현할 수 있다.

② 프로그램 실행(Program execution) : 프로그램을 실행하려면 먼저 메모리에 적재해야 하고, 프로세서 시간을 할당해야 한다.

③ 입출력 동작(I/O operations) : 수행 중인 프로그램은 입력이 필요하며, 사용자가 제공하는 입력을 처리한 후에는 출력을 생성해야 한다.

④ 파일시스템 조작(File-system manipulation): 사용자는 디스크에서 파일을 열고, 파일을 저장하며, 파일을 삭제하는 등 다양한 파일 조작 작업을 한다.

⑤ 통신(Communications) : 다중 작업 환경에서 공유 메모리를 이용하거나 메시지 전달로

다양한 유형의 프로세스와 통신을 지원한다.

⑥ 오류 탐지(Error detection) : 하드웨어와 소프트웨어 수준에서 오류를 탐지하고, 시스템을 모니터링하여 조정함으로써 하드웨어 문제를 예방한다.

43. 시스템 호출의 개념과 시스템 호출 방법을 설명하시오.

- 실행 중인 프로그램과 운영체제 사이의 인터페이스로 API(Application Programming Interfaces)라고도 한다. 시스템 호출에는 핵심 커널 서비스와 통신, 새로운 프로세스의 생성과 실행, 하드웨어 관련 서비스 등이 있다.
- 시스템 호출 방법 : 첫 번째는 프로그램에서 명령이나 서브루틴의 호출 형태로 호출하는 방법, 두 번째는 시스템에서 명령 해석기를 사용하여 대화 형태로 호출하는 방법

44. 계층 구조 운영체제의 장점은?

운영체제를 다수의 계층(수준)들로 분할하여 구성(시스템 모듈화)하여 시스템 검증과 오류 수정을 쉽게 할 수 있으며 모듈화로 시스템 설계나 구현이 단순해진다.

45. 단일 커널 구조 운영체제를 설명하고, 장단점을 기술하시오.

가장 보편적인 형태로 작고 간단하면서 시스템 기능이 제한된 구조로 모든 기능을 커널과 동일한 메모리 공간에 적재한 후 시스템 호출만으로 사용한다.

[장점]

대부분의 기능을 커널(시스템 운영에 필요한 많은 서비스 루틴을 커널 하나에 포함)에 그룹화해서 구현하여(구현이 간단) 시스템 자원을 효율적으로 관리할 수 있다.

[단점]

커널 크기가 커지면서 버그의 원인이나 기타 오류 구분이 어렵다, 새 기능 추가와 수정 및 유지 보수가 어렵다, 동일한 메모리에서 실행하므로 한 부분에서 발생한 문제가 시스템 전체에 영향을 준다.

46. 마이크로 커널 구조 운영체제를 설명하고 장단점을 기술하시오.

커널에는 최소 기능만 포함시켜 크기를 대폭 줄이고 기타 기능은 사용자 공간으로 옮겨 사용자 영역에서 수행하는 서버 구현 방식

[장점]

- 운영 체제 확장 용이성(서버 개발 용이)과 높은 이식성(기능 변경 용이)
- 서버에서 잘못 수행이 다른 서버와 커널에 영향 없다(보안과 신뢰성 제공).

[단점]

- 시스템 기능(모듈 간에 통신 증가)의 오버 헤드로 인한 성능 감소
- 응용 프로그램과 서버 간에 자료를 교환(문맥 전환)으로 속도가 느리다.

47. 시스템 및 응용 프로그램의 차이를 기술하시오.

- 시스템 프로그램 : 커널의 일부가 아니지만 운영 체제와 연관된 프로그램으로 컴퓨터 자원을 관리하고 응용 프로그램의 실행을 지원하여 컴퓨터를 제어하는 프로그램. 운영체제를 비롯해 장치 드라이버 등으로 구성된다.
- 응용 프로그램 : 시스템의 작동과 연관되지 않으나 특정 작업을 수행하려는 목적, 즉 어떤

문제를 해결하려고 사용자나 전문가가 만든 프로그램. 웹 브라우저, 워드 프로세서, 게임, 이미지 편집 프로그램, 은행 시스템 및 항공 예약 시스템 등이 해당한다.

Chapter 3 프로세스와 스레드_연습문제

1	①	**11**	①	**21**	③	**31**	③
2	④	**12**	①	**22**	③	**32**	③
3	①	**13**	③	**23**	③	**33**	①
4	①	**14**	④	**24**	①	**34**	④
5	①	**15**	①	**25**	④	**35**	④
6	③	**16**	④	**26**	②	**36**	③
7	②	**17**	②	**27**	③	**37**	②
8	④	**18**	④	**28**	③		
9	③	**19**	①	**29**	①		
10	①	**20**	④	**30**	③		

38. 프로세스를 사용자 관점과 시스템 관점으로 구분하여 설명하시오.

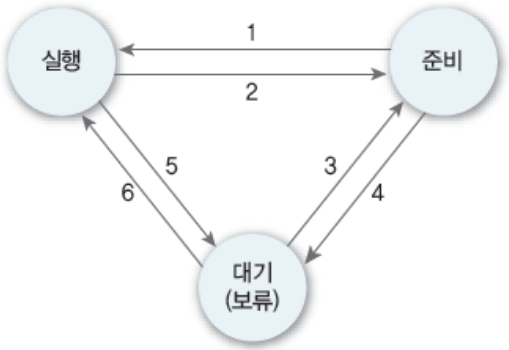
- [그림 3-2]과 [그림 3-3] 참고
- 사용자 관점 : 프로그램 카운터나 레지스터와 같이 현재 사용되는 자원에 대한 정보를 가지는 능동적인 개체로 세그먼트(코드, 데이터, 스택 등)의 가상 주소 공간을 가지며 실행되는 프로그램
- 시스템 관점 : 실행 중인 프로그램으로 실행 순서를 결정하는 스케줄러가 디스크에 저장된 프로그램에 프로세서를 할당해 실행 프로세스가 되어 장치나 메모리 같은 파일과 관련된 자원을 참조하고, 프로세스 지원과 협력을 위해 교착 상태•보호•동기화 같은 정보 교환.

39. 메모리에 있는 프로세스 주소 공간의 구성 요소와 그 역할을 설명하시오.

- [그림 3-2] 프로세스의 일반적인 메모리 구조(사용자 관점의 프로세스) 참고
- 스택(Stack) : 호출된 함수의 반환 주소, 반환값, 매개변수 등에 사용된다. 함수를 호출할수록 커지고 반환하면 줄어든다. 데이터를 일시적으로 저장하는 영역으로 지역 변수를 위해 사용된다.
- 힙(Heap) : 텍스트(코드) 영역과는 별도로 유지되는 자유 영역이다. 동적 메모리 할당을 위해 프로그램 실행 중 시스템 호출를 통해 사용되다가 해제하는 방법으로 사용한다. 동적 메모리 할당이 발생하면 보통 위쪽으로 커진다.
- 데이터 : 프로그램의 가상 주소 공간으로, 전역 변수나 정적 변수를 저장하거나 할당하고 실행하기 전에 초기화된다.
- 코드 : 프로그램이 시작될 때 프로세서가 디스크에서 읽어 실행하는, 컴파일된 프로그램을 저장한다. 프로세스에 의해 변경 될 수 없고 읽기 전용으로 프로그램이 코드 영역을 침범하여 쓰기를 시도하면 오류를 발생시켜 이 프로그램이 종료되게 한다.

40. 프로세스가 그림과 같이 3상태 사이를 각 화살표처럼 이동하는 원인을 간단히 설명하시

오. 발생(이동)하지 않으면 N/A로 표시하시오.



- 화살표 1 : 프로세스는 예약 스케줄러(디스패치)에 의해 실행된다.
- 화살표 2 : 일정 시간이 지나 실행 중단되나 프로세스는 실행하기 위해 여전히 기다린다(양보).
- 화살표 3 : 입출력 완료(세마포어에 의해 깨어 남)
- 화살표 4 : N/A
- 화살표 5 : 입출력 요청, 자원 요청.
- 화살표 6 : N/A

41. 프로세스 중단(서스펜드)과 재시작은 언제 발생하는가?

- 시스템에 장애가 발생하면 실행 중인 프로세스는 잠시 중단했다가, 시스템이 기능을 회복할 때 다시 재시작할 수 있다.
- 프로세스에 의심스러운 부분이 있으면 실행 중인 프로세스를 중단하여 확인한 후 재시작하거나 정지할 수 있다.
- 처리할 작업이 너무 많아 시스템 부담이 크면(너무 많은 적재) 프로세스 몇 개를 중단했다가 시스템이 정상 상태로 다시 돌아왔을 때 재시작할 수 있다.

42. 프로세스 제어 블록PCB에 저장하는 정보는?

- [그림 3-7] 프로세스 제어 블록 참고
- 프로세스에 대한 특정 정보를 저장하는 데이터 블록이나 레코드로 작업 제어 블록(TCB)이라고도 한다.

43. 프로세스에는 스레드가 하나 이상 있으며, 병렬로 수행된다. 스레드를 이용하면 얻는 이점을 기술하시오.

- 사용자에 대한 응답성 증가
- 프로세스의 자원과 메모리 공유로 시스템 성능 향상
- 경제성 향상(오버헤드 감소)
- 멀티 프로세싱을 통해 성능과 효율 향상

44. 프로세스의 3상태 변화 그림을 그리고, 각 상태를 간략하게 설명하시오.

- [그림 3-5] 프로세스 상태 변화 참고
- 실행 : 명령어가 실행되는 상태, 즉 프로세서를 점유한 상태

- 대기(보류) : 프로세서가 어떤 이벤트(입출력 종료와 같은 외부 신호)가 일어나기를 기다리는 상태
- 준비 : 프로세서에 할당되기 기다리는 상태

45. 사용자 수준 스레드와 커널 수준 스레드의 차이는? 그리고 각 구현 방법은 언제 더 효과적인가?

① 사용자 스레드는 스레드 라이브러리를 이용해 커널의 지원 없이 생성, 스케줄링, 관리 등을 할 수 있다. 스레드의 이런 동작은 모두 커널의 지원 없이 사용자 공간에서 이루어진다. 사용자 스레드는 프로세스 내부에 스레드 여러 개로 구성될 수 있고, 한 프로세스에서 같은 자원을 가지고 다른 사용자 스레드와 통신할 수 있으니 당연히 문맥교환으로 인한 오버헤드가 없다.
  - 커널을 지원하지 못함
  - OS에 알려지지 않고 사용자 공간에서 실행되는 라이브러리에 의해 생성되고 관리됨
  - 다른 스레드로의 모드 전환이 불필요하여 효율적
  - 한 스레드 보류로 전체 프로세스가 보류됨

② 커널 스레드는 사용자 스레드서 LWP 자료구조를 통해 연결되는데, 보통 커널 스케줄러에 의해 프로세서를 할당받아 사용하지만, 직접 할당 받을 수도 있어 효율적. 하지만 사용자 수준 스레드보다 속도가 느리다.
  - 같은 프로세스의 다중 스레드가 병행 실행
  - 한 스레드의 보류가 다른 스레드를 스케줄링
  - 문맥교환을 요구하여 오버헤드 발생

46. 스레드를 생성할 때 사용하는 자원은? 이것은 프로세스를 생성할 때와 어떻게 다른가?

프로세스가 작업을 수행하려면 프로세서 시간, 메모리, 입출력 장치 등의 자원이 필요하지만 스레드가 생성될 때는 프로그램 카운터, 레지스터, 스택 제어권 자원만 필요하다. 운영체제는 프로세스와 스레드의 개념을 분리(프로세스는 자원 소유 단위, 스레드는 실행 단위)하여 프로세스 하나의 실행 단위 다수가 자원을 공유할 수 있어 자원 생성과 관리 중복성을 최소화하여 수행 능력을 향상시킨다. 프로세스는 명령과 데이터를 할당할 메모리와 코드도 메모리에 할당되도록 적재한다. 스레드는 프로세스보다 작기 때문에 프로세스 생성보다 적은 자원을 사용한다. 프로세스 하나의 생성은 대단위 데이터 구조보다 PCB 할당이 요구된다. PCB는 개방 파일 목록, 환경 변수, 메모리 맵을 포함한다. 메모리 맵 운영과 할당은 대부분 시간이 걸리는 활동이다. 사용자, 커널 스레드 생성은 레지스터 셋, 스택, 우선순위 유지를 위한 작은 데이터 구조 할당을 포함한다.

47. 프로세스를 종료하는 과정을 예로 들어 설명하시오.

- 정상적인 완료 : 프로세스가 운영체제의 서비스를 호출한 경우
- 시간 초과 : 프로세스가 명시된 전체 시간을 초과하면서 실행되거나 명시된 시간을 초과하면서 어떤 이벤트 발생을 기다리는 경우
- 실패 : 파일 검색 실패, 명시된 횟수를 초과하여 입출력이 실패한 경우
- 산술 오류, 보호 오류, 데이터 오류 등
- 메모리 부족, 액세스 위반 등

48. 인터럽트와 트랩의 차이를 설명하시오.

- 인터럽트 : 현재 실행되는 프로세스와 별도로 외부에서 발생되는 여러 종류의 이벤트(예를 들면 입출력 동작의 종료)에 의해 발생한다.
  - 입출력 인터럽트 : 입출력 동작의 발생을 확인한 후 이벤트를 기다리는 프로세스를 준비 상태로 바꾸고 실행할 프로세스를 결정한다.
  - 클록 인터럽트 : 현재 실행 중인 프로세스의 할당 시간을 조사하여 프로세스를 준비 상태로 바꾸고 다른 프로세스를 디스패치한다.
  - 메모리(페이지) 부재 : 메모리의 페이지 부재로 교체되기를 기다림(8장 참고)
- 트랩(trap) : 부적절한 파일 접근이나 현재 실행 중인 프로세스에 의해 발생되는 오류나 예외 상황 때문에 발생한다.

49. 문맥 교환과 발생하는 시기를 기술하시오.

수행 중인 프로세스를 다른 프로세스로 전환하려면 이전의 프로세스 상태 레지스터 내용을 보관하고, 또 다른 프로세스의 레지스터를 로드해야 하는데, 이런 일련의 과정을 문맥교환(Context Switching) 또는 프로세스 교환이라 하며 오버헤드가 발생하고, 이는 메모리 속도, 레지스터 수, 특수 명령어의 존재에 따라 달라진다. 문맥교환은 프로세스가 '준비 → 실행' 상태로 변하거나 '실행 → 준비', 또는 '실행 → 대기' 상태로 변할 때 발생한다.

50. 사용자 수준 스레드의 장단점을 설명하시오.

[장점]

- 오버헤드 감소
- 스케줄링의 유연성
- 높은 이식성

[단점]

- 시스템의 동시성 지원 불가
- 시스템 규모 확장 제약
- 스레드 간 보호가 어려움

51. 마이크로 커널 구조 운영체제를 설명하고, 장단점을 기술하시오.

커널에는 최소 기능만 포함시켜 크기를 대폭 줄이고 기타 기능은 사용자 공간으로 옮겨 사용자 영역에서 수행하는 서버 구현 방식

[장점]

- 운영 체제 확장 용이성(서버 개발 용이)과 높은 이식성(기능 변경 용이)
- 서버에서 잘못 수행이 다른 서버와 커널에 영향 없다(보안과 신뢰성 제공).

[단점]

- 시스템 기능(모듈 간에 통신 증가)의 오버 헤드로 인한 성능 감소
- 응용 프로그램과 서버 간에 자료를 교환(문맥 전환)으로 속도가 느리다.

52. 동일한 프로세스의 스레드 사이에서 ① 스택 ② 데이터 세그먼트 ③ 힙 중 무엇을 공유하는가?

데이터 세그먼트와 힙(heap)이 공유된다.

53. 프로세스 제어 블록의 목적과 운영체제는 언제 업데이트하고 판독하는가?

주어진 프로세스의 자원과 프로세서(레지스터 변수)의 상태를 추적한다. 프로세스가 선점(프로세스 교환)될 때 업데이트된다. 프로세스가 다시 저장될 때(프로세스 교환) 판독한다.

54. 일대일(1:1) 스레딩 모델과 다대일(n:1) 스레딩 모델의 차이점과 일대일(1:1) 스레딩 모델을 선호하는 이유를 설명하시오.

이러한 모델들은 커널 스레드 당 사용자 영역 스레드의 수를 정의한다. 1:1은 각각의 사용자 영역 스레드에 대한 하나의 커널 스레드가 있다. N:1은 하나의 커널 스레드에 모든 사용자 영역 스레드를 가지고 있다. 1:1 모델은 각각 스케줄 커널 스레드에 다른 사용자 영역 스레드가 대응하기 때문에 실행되고 동시에 사용자 영역 스레드의 차단을 허용하므로 N:1에서 겪는 커널 지원 부족 문제(시스템의 동시성 지원 불가)와 사용자 영역 스레드의 시스템 호출로 인한 다른 스레드의 중단 문제를 해결할 수 있다.

55. 프로세스와 스레드의 차이를 설명하시오.

프로세스는 프로세스와 관련된 주소 공간을 가지고 항상 운영체제에 의해 스케줄링되고 표시된다. 스레드들은 동일한 주소 공간을 공유하며 사용자 공간에서 스케줄링 할 수 있다.

Chapter 4 병행 프로세스와 상호배제_연습문제

1	③	**11**	②	**21**	②
2	②	**12**	④	**22**	④
3	①	**13**	②	**23**	③
4	①	**14**	③	**24**	③
5	③	**15**	①	**25**	③
6	④	**16**	②		
7	①	**17**	①		
8	④	**18**	③		
9	④	**19**	④		
10	③	**20**	②		

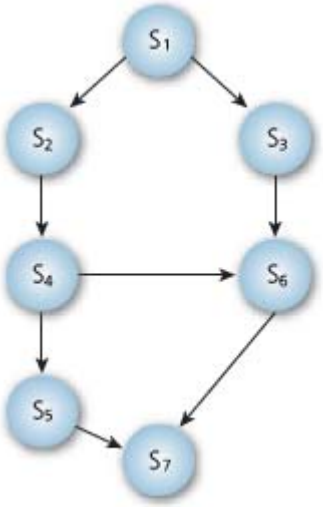
26. 바쁜 대기의 개념과 이를 피할 수 있는 방법을 간단히 설명하시오.

- 바쁜 대기 : 한 프로세스가 임계영역에 있으면 이 임계영역에 들어가려는 프로세스는 진입코드에서 계속 반복 순환하는 것으로 다중 프로그래밍에서 문제가 된다.
- 바쁜 대기를 피하는 방법 : 세마포어를 사용하여 피할 수도 있다. 세마포어는 다양한 동기화 문제들을 해결하기 위해 사용할 수 있으며 효과적으로 구현할 수 있다. 바쁜 대기 상황은 wait와 signal 세마포어 연산을 수정하면 된다. wait 연산을 수정하였을 때 세마포어 값이 양수가 아니면 스스로 프로세스의 상태를 대기 상태로 바꾼다. 세마포어 S에 의해 대기 또는 중지된 프로세스는 다른 프로세스의 signal 연산에 의해 재시작 될 수 있다.

27. 대기의 종류를 세 가지 이상 나열하고 간단히 설명하시오.

- 제한된 대기: 한 프로세스가 임계영역에 대한 요청 후부터 요청이 수락되기까지의 기간 내에 다른 프로세스가 임계영역을 수행할 수 있는 횟수에는 제한이 있어야 한다는 조건.
- 무한 대기: 상대방 프로세스의 상태를 모르고 실행을 멈추고 계속하여 기다리고 있는 상태를 말한다.
- 점유와 대기: 적어도 하나의 자원을 보유하고 현재 다른 프로세스에 할당된 자원을 얻기 위해 기다리는 프로세스가 있어야 한다.
- 순환대기: 프로세스들이 순환을 이루어서 존재, 이를 구성하는 각 프로세스는 순환내의 이전 프로세스가 요청하는 자원을 점유하고 다음 프로세스가 점유하고 있는 자원을 요구

28. 다음 선행 그래프를 fork/join 문장으로 구현하시오.




```
 S1
 count1:=2;
 fork  L1;
 S2
 S4
 count2:=2;
 fork  L2;
 S5;
 Go  to  L3;
L1: S3
L2: join  count1;
   S6
L3: join  count2;
   S7
```

29. 버퍼 크기가 유한한 생산자·소비자 문제에서 버퍼가 최대 n-1개의 데이터만 허용하는 이유를 설명하시오.

버퍼가 꽉차있거나 비워있을 경우 버퍼의 접근을 배제해야 하기 때문이다.

30. 임계 영역 문제에서 세마포가 아닌 TestAndSet을 사용하여 해결할 때의 단점은?

TestAndSet 사용은 바쁜 대기이다. 다른 프로세스가 lock(즉 0으로 잠금 설정) 해제까지 임계영역 밖에서 대기하므로(진입 조건이 true가 될 때까지 반복적인 조사로 바쁜 대기) 프로세서 시간이 낭비된다. 세마포 사용은 바쁜 대기가 발생되지 않는다. 세마포 값이 0이면 Wait(S) 연산을 수행하여 프로세스는 차단된다. 차단 된 프로세스는 다른 프로세스가 세마포어에 Signal(S) 연산을 수행 할 때까지 실행을 재개하지 않으므로 바쁜 대기를 회피(방지)할 수 있다.

31. 임계 영역의 의미와 문제점을 간단히 설명하시오.

임계영역은 한 순간에는 하나의 프로세스만 사용할 수 있는 영역(공유 자원의 독점을 보장하는 코드 영역)을 의미한다. 임계영역의 문제점은 한순간에 교착 상태 없이 하나의 프로세스가 임계영역 진입하도록 설계하는 알고리즘

32. 임계 영역이 만족해야 하는 세 가지 조건을 기술하시오.

상호배제, 진행, 제한된 대기

33. 프로세스들의 병행 처리를 명시하는 언어적 표현을 간략하게 기술하시오.

fork/join에 의한 방법과 parbegin/parend를 사용하는 방법이 있다.
[fork/join]
fork L은 L이라는 레이블이 붙은 문장과 fork L 바로 다음 문장을 병행 수행하라는 의미이고, join count는 2개 이상의 병행문장들을 하나로 재결합시키는 것으로 어떤 문장의 처리가 끝이 나서 join문을 만나더라도 나머지 문장들은 계속 수행되어야 함을 의미하는데, 여기서 count는 병행 처리할 문장의 개수를 의미하고 join문을 만날 때마다 1 감소하게 되며 count가 0이 되면 비로소 대기를 끝내고 다음 문장을 실행하게 된다.

[parbegin/parend]
parbegin과 parend 사이의 모든 문장은 병행하여 수행될 수 있다. 병행문장은 현대의 블록 구조 고급 언어에 쉽게 덧붙일 수 있고 다른 구조적 제어 문장의 장점을 많이 보여 준다. 세마포어의 개념을 도입해야만 fork/join과 같은 능력을 가지게 된다.

34. 세마포의 두 가지 연산을 설명하시오.

세마포어(S)는 P와 V라는 표준 연산에 의해서만 접근되어지는 정수형 변수로써, S는 표준단위 P(프로세스를 대기시키는 wait 동작으로 임계영역에 진입하기 위한 동작) 연산과 V(대기중인 프로세스를 깨우는 신호를 보내는 signal 동작으로 임계영역에서 나오기 위한 동작) 연산에 의해서만 접근되는 정수 변수다. P와 V의 정의는 다음과 같다.

P(S) : while S ≤ O do no-op;
S := S - 1;
V(S) : S := S + 1;

표준 연산 P와 V의 연산 조건은 하나의 프로세스가 세마포어 S의 값을 수정할 때에는 다른 프로세스가 세마포어 S의 값을 수정할 수 없다는 것이다. 그리고, 특히 P(S) 연산의 경우에는 (S ≤ 0)에 대한 조건 검사와 (S := S - 1)의 수행은 인터럽트 없이 수행된다는 조건을 갖는다.

35. 세마포의 단점을 설명하시오.

한 개의 프로세스가 임계 구역에 있는 동안, 임계 구역에 진입하려는 다른 모든 프로세스가 진입 영역에서 계속 대기한다.
- 바쁜 대기 “while S ≤ 0 do skip ;” while 조건을 계속 검사

36. 모니터의 개념을 설명하시오.

핸슨(Brich Hansen)과 호(Hoare)에 의해 적당한 동기화를 자동적으로 제공하는 모듈로 프로그래밍 언어에서 제공되는 프로그래머-정의 연산자들의 집합으로 구성된다.
[그림 4-17]과 같이 하나 또는 그 이상의 프로시저(연산 동작들)와 초기화 코드, 그리고 공유 데이터로 구성된 소프트웨어 모듈로 이루어진 객체다. 모니터는 한 순간에 하나의 프로세스만 모니터 안에서 활동하도록 보장한다.

37. 경쟁 상태를 설명하고, 이를 피할 수 있는 방법을 간단히 설명하시오.

경쟁 상태는 여러 프로세스가 공유 데이터를 동시에 접근할 때 공유 데이터에 대한 접근 순서에 따라 실행 결과가 달라지는 상황이다. 경쟁 상태를 방지하기 위해 병행 프로세스를 동기화해야하며 임계 영역을 이용한 상호배제를 통해 해결할 수 있다.

38. 세마포와 조건 변수의 wait와 signal 연산 차이를 설명하시오.

세마포에서 wai 연산은 0 이하의 경우(S <= 0) 차단된다. 진행되면 하나의 값을 감소(S--)시키며 signal 연산은 무조건 하나의 값을 증가(S++)시킨다. 모니터의 조건 변수의 경우, wait 연산은 다른 프로세스에 의해 깨울 때까지(signal 호출) 프로세스 실행을 무조건 차단한다. signal 연산은 차단 된 프로세스를 깨우고(다시 시작) 그렇지 않으면(기다리는 프로세스가 없으면) 아무것도하지 않는다.

Chapter 5 교착 상태와 기아 상태_연습문제

1	①	**11**	②	**21**	④
2	①	**12**	①	**22**	④
3	①	**13**	④	**23**	②
4	④	**14**	①	**24**	②
5	③	**15**	②	**25**	①
6	①	**16**	②	**26**	②
7	③	**17**	②	**27**	④
8	④	**18**	④	**28**	②
9	④	**19**	④		
10	①	**20**	③		

29. 하벤더의 교착 상태 예방 연구에서 기본 전제가 되는 것은?

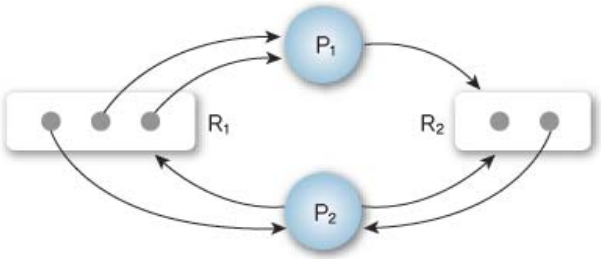
교착 상태가 발생하기 위한 필요 조건 중 어느 하나라도 배제한다면 시스템에서 교착 상태가 발생할 수 없다.

30. 점유와 대기 조건의 방지 방법과 문제점은?

점유와 대기 조건이 시스템에 발생하지 않으려면 1) 프로세스가 작업을 수행하기 전에 필요한 모든 자원을 요청하여 획득한다. 2) 프로세스가 자원을 전혀 갖고 있지 않을 때만 자원을 요청할 수 있도록 허용해야 한다.

문제점은 1) 자원의 효율성이 너무 낮다. 2) 기아 상태가 발생 가능성이 있다.

31. 다음과 같은 자원 할당 그래프에 대한 물음에 답하시오.



① 시스템은 교착 상태에 있는가?

아니오, P_1과 P_2 모두 여전히 진행할 수 있다. 예를 들어, P_1과 P_2 모두 R_2 자원을 획득할 수 있다.

② P_2 자원에 P_1 요청을 먼저 제공할 때 시스템 상태는 안전한가?

안전하다. 자원이 승인되면 P_1은 어떤 자원도 기다리지 않기 때문에 실행할 수 있다. 종료 후 P_1은 자원을 해제하면 P_2은 진행을 할 수 있다.

③ P_2 자원에 P_2 요청을 먼저 제공할 때 시스템 상태는?

교착 상태. P_1과 P_2 모두 다른 하나가 보유하고 있는 자원을 기다리고 있을 것이다.

32. 교착 상태를 회복하는 해결 방법을 간단히 설명하시오.

- 순환 대기를 탈피하기 위하여 단순하게 한 개 이상의 프로세스 중지
- 교착 상태의 프로세스들로부터 자원을 선점

33. 교착 상태를 벗어나려고 중단(종료)시킬 프로세스를 선정할 때 고려할 기준을 나열하시오.

- 프로세스들의 우선순위
- 지금까지 프로세스가 수행된 시간과 앞으로 종료하는데 필요한 시간
- 프로세스가 사용한 자원 형태와 수(예를 들어, 자원들을 선점할 수 있는지에 대한 여부)
- 프로세스 종료를 위하여 더 필요한 자원의 수
- 종료하는 데 필요한 프로세스의 수
- 프로세스가 대화식인지 일괄식인지의 여부

34. 컴퓨터 시스템 환경과 관련되지 않은 교착 상태의 실례를 세 가지만 들어 보시오.

① 자동차 한 대 만이 지나갈 수 있는 도로에서 자동차 두 대가 서로 반대 방향으로 가려는 하는 경우

② 사다리에서 한 사람이 내려오는 동안 다른 사람은 올라가려고 하는 경우.

③ 두 대의 기차가 각각 같은 트랙에서 반대 방향으로 진행(움직이는) 하고 있는 경우

35. 하벤더의 교착 상태를 예방하는 세 가지의 기본 방법은?

① 각 프로세스는 한 번에 자기가 필요한 모든 자원을 요구해야 하며, 요청한 자원을 모두 제공받기 전까지는 작업 진행을 할 수 없다.

② 어떤 자원을 점유하고 있는 프로세스의 요구가 더 이상 허용되지 않으면 점유한 자원을 모두 반납하고 필요할 때 다시 자원을 요구해야 한다.

③ 모든 프로세스에 자원을 순서대로 할당해야 한다. 모든 프로세스에 각 자원 유형별로 할당 순서를 부여한 후, 순서에 따라 자원을 요구하도록 한다.

교착 상태를 피하기 위한 자원 사용에 대한 정보로는, 해당 시스템이 갖고 있는 모든 자원의 종류와 각 자원의 최대 개수, 각 프로세스들이 필요로 하는 자원의 최대 요구량, 각 프로세스에게 이미 할당된 자원의 수, 각 프로세스들이 추가로 원하는 자원의 요구량, 각 자원의 사용 가능 개수 등이 있다.

36. 교착 상태와 기아 상태 사이의 주요 차이점은?

교착 상태는 모든 프로세스가 그 집합 내의 다른 프로세스에 의해서만 발생될 사건(event)을 기다리고 있는 상태를 말한다. 기아 상태는 시스템은 교착 상태는 아니나 적어도 한 프로세스는 불명확하게(막연히) 연기되는 상황이다.

37. 시스템 상태가 다음과 같다고 가정할 때 은행가 알고리즘을 이용하여 다음 물음에 답하시오.

프로세스	Allocation	Max	Available
	ABCD	ABC	ABCD
P_0	3011	5111	1020
P_1	0100	0212	
P_2	1110	4210	
P_3	1101	1111	
P_4	0000	2110	

① 배열 Need의 내용은?

Need
ABCD
2100
0112
3100
0010
2110

② 이 시스템은 안정 상태인가?

네, Available (1, 0, 2, 0) 이면 <P_3, P_4, P_0, P_1, P_2> 순서로 안정

③ 프로세스 P1에서 요청이 (0, 4, 2, 0)이면 이 요청은 즉각 받아들여질 수 있는가?

네,

i. (1, 0, 1, 0) ≤ Available = (1, 0, 2, 0)

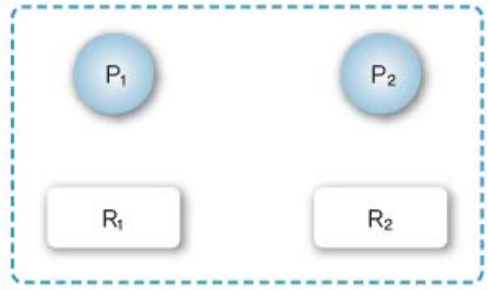
ii. (1, 0, 1, 0) ≤ Max = (2, 1, 1, 0)

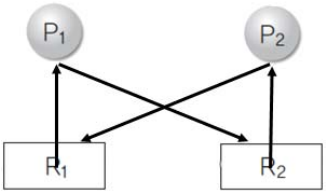
iii. 할당 후 새로운 시스템 상태

프로세스	Allocation	Max	Need	Available
	ABCD	ABCD	ABCD	ABCD
P_0	3011	5111	2100	0010
P_1	0100	0212	0112	
P_2	1110	4210	3100	
P_3	1101	1111	0010	
P_4	1010	2110	1100	

따라서 <P_3, P_4, P_0, P_1, P_2> 순서로 안정

38. 프로세스 2개와 자원 2개가 교착 상태에 있다는 것을 나타내도록 자원 할당 그래프를 그리시오.





39. 경쟁 상태와 교착 상태의 차이점은?

경쟁 상태는 동기화 없이 공유 자원(상태)에 액세스하려는 하나 이상의 프로세스에 의해 발생한다. 프로세스는 계속 실행되므로 프로그램은 잘못된 결과를 가질 수 있다. 교착 상태에서 프로세스는 진행 할 수 없다. 프로세스들은 모두 다른 세 가지 교착 상태 조건이 유지되고 있는 동안 서로 보유 자원(원형 대기)을 획득하기 위해 대기하고 있다.

40. 자원 할당 그래프에 주기가 있는 시스템을 가정한다. 이때 교착 상태라고 정의(단정)할 수 있는가?

아니오, 교착 상태라고 결정적으로 인정할 수 없다. 주기(원형) 외에 교착 상태 발생에 필요한 다른 3 조건(상호배제, 점유와 대기, 선점)이 필요하다. 상기 조건 중 하나가 만족되지 않으면, 교착 상태가 될 수 없다.

Chapter 6 프로세스 스케줄링_연습문제

1	③	**16**	④	**31**	①	**46**	④
2	①	**17**	③	**32**	②	**47**	①
3	④	**18**	④	**33**	③	**48**	②
4	②	**19**	②	**34**	③	**49**	④
5	④	**20**	④	**35**	①	**50**	②
6	③	**21**	④	**36**	②	**51**	④
7	③	**22**	①	**37**	②	**52**	④
8	④	**23**	①	**38**	①	**53**	③
9	①	**24**	④	**39**	④	**54**	③
10	③	**25**	②	**40**	③	**55**	①
11	④	**26**	③	**41**	②	**56**	③
12	③	**27**	①	**42**	③	**57**	①
13	②	**28**	④	**43**	③	**58**	①
14	②	**29**	④	**44**	③	**59**	②
15	①	**30**	③	**45**	①		

60. 단기 스케줄링, 중기 스케줄링, 장기 스케줄링의 차이를 기술하시오.

- 단기 스케줄링 : 프로세서(CPU) 스케줄러라고 부르며 메인 메모리의 준비상태에 있는 작업 중에서 실행할 작업을 선택하고 프로세서를 배당하는 일을 한다.
- 중기 스케줄링 : 현재 생성되어 있는 프로세스 중에 비효율적으로 시스템의 자원을 낭비하고 있는 프로세스가 있을 경우 보조기억장치로 추방하는 스케줄링이다. 즉 교체 기능의 일부로 메인 메모리에서 부분적인 적재가 이루어지고 일시 중지의 원인이 해결되면 다시 준비상태가 된다.
- 장기 스케줄링 : 작업 스케줄러라고 부르기도 하며 어떤 작업이 시스템에 들어와서 스케줄링 원칙에 따라 디스크 내의 어떤 작업을 어떤 순서로 메모리에 가져와서 처리할 것인가를 결정하는 프로그램

61. 선점 스케줄링과 비선점 스케줄링의 차이점을 정의하시오. 엄격한 비선점식 스케줄링을 사용하지 않는 이유도 설명하시오.

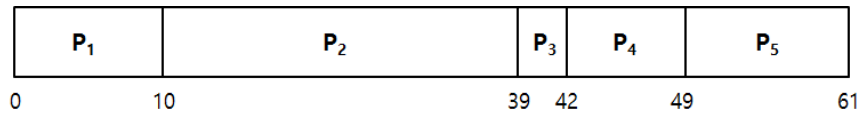
- 선점 스케줄링: 실행중인 프로세스를 중간에 중지하고 다른 프로세스를 실행할수 있음
- 비선점 스케줄링: 실행중인 프로세스를 중지 불가능함

62. 다음 프로세스들이 시간 0에 P_1, P_2, P_3, P_4, P_5 순으로 도착한다고 가정하여 다음 질문에 답하시오.

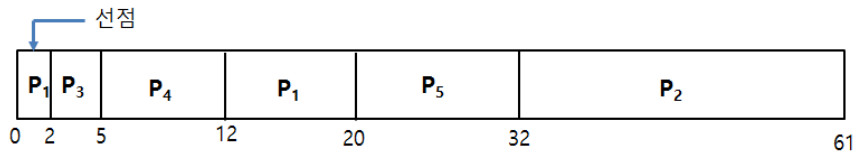
프로세스	버스트 시간 (단위 : 밀리초)	우선순위 (숫자가 작을수록 우선순위가 높음)
P_1	10	3
P_2	1	1
P_3	2	3
P_4	1	4
P_5	5	2

① 선입선처리, 최소작업 우선, 비선점 우선순위, 순환 할당(할당량=1)을 이용하여 이를 프로세스들의 실행을 설명하는 간트 차트로 그리시오.

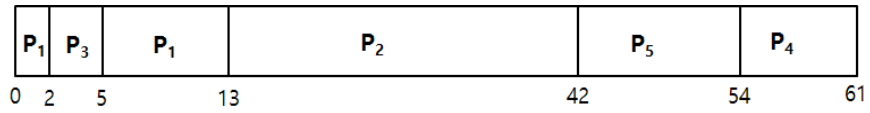
[선입선처리]



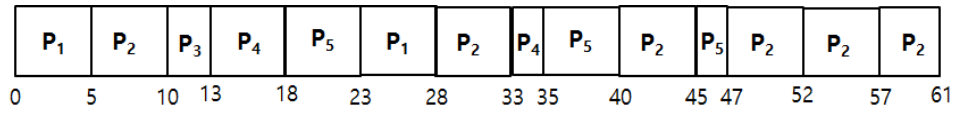
[최소작업 우선]



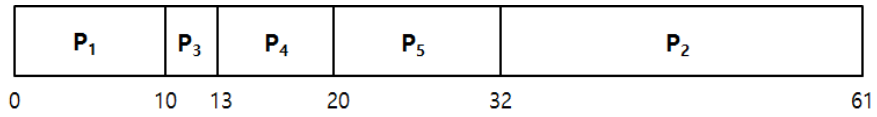
[선점 우선순위]



순환 할당



HRN



② 각 스케줄링 알고리즘에 대한 반환시간은?

선입선처리 : 38.2(=(10+38+40+46+57)/5)

선점 최소작업 우선 : 24(= (20+60+3+9+28)/5)

선점 우선순위 : 33(=(13+41+3+58+50)/5)

순환 할당 : 34.8(=(28+60+11+32+43)/5)

HRN : 25.2(=(10+60+11+17+28)/5)

③ 각 스케줄링 알고리즘에 대한 대기시간은?

선입선처리 : 26(=(0+9+37+39+45)/5)
선점 최소작업 우선 : 11.8(=(10+31+0+2+16)/5)
선점 우선순위 : 20.8(=(3+12+0+51+38)/5)
순환 할당 : 22.6(=(18+31+8+25+31)/5)
HRN : 12.8(=(0+31+8+10+16)/5)

④ 어떤 스케줄링이 모든 프로세스에서 최소의 평균 대기시간을 갖는가?
평균 대기시간 11.8로 선점 최소작업 우선 방식의 평균대기시간이 가장 작다.

63. 스케줄링의 목적을 기술하시오.
- 자원 할당의 공정성 보장
- 단위 시간당 처리량 최대화
- 오버헤드 최소화
- 자원 사용의 균형 유지 등 본문 참조

64. 스케줄링의 성능 기준 요소를 기술하시오.
- 사용률과 처리율 최대화
- 반환시간 최소화
- 대기시간 최소화
- 반응 시간 최소화

65. 다단계 피드백 큐 스케줄링 알고리즘과 전면 작업에는 라운드 로빈(순환 할당) 스케줄링을 사용하고, 후면 작업에는 선점 우선순위 알고리즘을 사용하는 다단계 큐(전면-후면) 프로세서 스케줄링 알고리즘의 차이를 설명하시오.
[다단계 귀환 스케줄링]
- 작업이 큐 사이를 이동 가능
- 서로 다른 프로세서 버스트 특성에 따라 분리 구분
- 작업이 요구하는 프로세서 시간이 너무 크면 낮은 단계 큐로 이동
- 입출력 중심작업과 전면 작업을 높은 우선순위 큐로 이동
- 낮은 우선순위 큐에서 오래 기다린 작업은 높은 우선순위 큐로 이동

[다단계 큐]
- 작업이 한 큐에만 고정되어 실행되며 큐 사이에 옮겨지지 않음
- 전면 작업과 후면 작업의 성질을 바꿀 수 없음
- 스케줄링 부담이 적은 장점이 있으나 융통성이 적음

Chapter 7 메모리 관리_연습문제

1	④	**16**	②	**31**	①	**46**	④
2	③	**17**	③	**32**	④	**47**	③
3	①	**18**	③	**33**	④	**48**	④
4	③	**19**	④	**34**	③	**49**	②
5	②	**20**	②	**35**	②	**50**	④
6	②	**21**	③	**36**	②	**51**	①
7	①	**22**	①	**37**	③		
8	②	**23**	④	**38**	①		
9	③	**24**	①	**39**	③		
10	③	**25**	②	**40**	①		
11	④	**26**	②	**41**	①		
12	④	**27**	②	**42**	②		
13	①	**28**	①	**43**	①		
14	④	**29**	④	**44**	③		
15	④	**30**	③	**45**	③		

52. 논리적 주소와 물리적 주소 차이를 설명하시오.

논리주소는 메모리 위치를 참조하며 메모리의 위치가 0과 관계된 사용자 프로그램에 의해 만들어지며 메모리에 있는 프로세스 코드와 데이터의 현재 위치와 별개이다. 반드시 현재 물리적 주소로 변환되어야 한다.

물리 주소는 메모리에 저장하거나 인출(Fetch) 데이터에 사용되는 절대 주소 또는 실제 주소다.

53. 프레임 32개로 구성된 실제 물리적 주소 공간으로 매핑되는 1,024바이트의 페이지 8개인 논리적 주소 공간을 생각해 보자.

① 논리적 주소에 몇 개의 비트가 있는가?

13비트. 1024바이트는 10bit로 나타낼 수 있다. (2^{10} =1024) 그리고 8페이지는 2^3 으로 나타낼 수 있으므로 3비트가 사용된다. 그러므로 10+3=13, 논리 주소는 13비트가 사용된다.

② 물리적 주소에 몇 개의 비트가 있는가?

15비트. 물리적 주소는 32프레임으로 구성되어 있으므로 ($32=2^5$) 5비트가 사용된다. 그러므로 10+5=15비트가 사용된다.

54. 다음 할당 알고리즘을 설명하시오.

① 최초 적합

사용 가능 공간 리스트에서 충분히 큰 첫 번째 공백 분할 공간에 할당한다.

② 최적 적합
사용 가능 공간 중에서 가장 작은 크기의 사용 공간을 작업에 할당한다.

③ 최악 적합
사용 가능 공간 중에서 가장 큰 사용가능공간에 작업을 할당한다. 작업에 할당하고 남은 블록은은 최상 적합에 의해 만들어진 것보다 더 크고 잠재적으로 유용할 것이라는 예상이다

④ 100KB, 500KB, 200KB, 300KB, 600KB 순으로 기억장치가 분할되었을 때 ①~③의 알고리즘은 프로세스 212KB, 417KB, 112KB, 426KB를 어떤 순서로 할당하는가? 어느 알고리즘이 기억장치를 가장 적절하게 사용하는가?
[최초 적합]
① 212KB는 500KB 분할에 할당된다.
② 417KB는 600KB 분할에 할당된다.
③ 112KB는 500KB 분할에 할당된다(새로 생긴 분할 288KB = 500KB - 212KB).
④ 426KB는 들어갈 공간이 없으므로 다른 process가 끝날 때까지 대기(wait)한다.

[최적 적합]
① 212KB는 300KB 분할에 할당된다.
② 417KB는 500KB 분할에 할당된다.
③ 112KB는 200KB 분할에 할당된다.
④ 426KB는 600KB 분할에 할당된다.

[최악 적합]
① 212KB는 600KB 분할에 할당된다.
② 417KB는 500KB 분할에 할당된다.
③ 112KB는 388KB 분할에 할당된다.
④ 426KB는 들어갈 공간이 없으므로 다른 process가 끝날 때까지 대기(Wait)한다.

위와 같은 예에서는 최적적합이 최상임을 나타내고 있다

55. 내부 단편화와 외부 단편화의 차이를 설명하시오.
- 내부 단편화 : 페이징에서 페이지 단위로 메모리에 적재시키는데 프로그램의 사이즈가 정해진 단위의 블록으로 나뉘다가 정해진 단위의 사이즈보다 조금 커져 조금 큰 부분을 하나의 블록에 저장하게 되어 블록 안에 생기는 빈 공간을 말한다.
- 외부 단편화 : 세그먼테이션에서 세그먼트 단위로 메모리에 적재시키는데 세그먼트와 세그먼트 사이의 공간이 다른 세그먼트는 들어갈 수 없을 정도의 빈 공간이 생기는 것을 말한다.

따라서 내부 단편화는 하나의 범위나 페이지 상의 영역으로 그 범위나 페이지를 차지하는 작업(프로세스)에는 사용되지 않는다. 물론 이 공간은 작업이 끝나거나 즉 범위나 페이지의

사용이 끝날 때까지는 그 시스템의 사용이 불가능하다

56. 왼쪽 세그먼트 테이블을 참고하여 오른쪽 ①~⑥의 논리적 주소에서 물리적 주소를 구하시오.

세그먼트	기본	길이
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

① 219 + 430 = 649

57. 세그먼트 하나가 다른 프로세스 2개의 주소 공간에 속할 수 있는 방법을 설명하시오.
세그먼트 테이블은 base-limit registers의 집합체이기 때문에 세그먼트들은 서로 다른 두 개의 일들을 나타내는 세그먼트 테이블안의 항목들이 똑같은 physical 위치를 나타낼 때 공유될 수 있다 그 두 개의 세그먼트 테이블은 동일한 base pointers를 가져야만 하고 공유된 세그먼트 수는 그 두 개의 과정에서 똑같아야 한다.

58. 페이지 테이블을 메모리에 저장한 페이징 시스템이 있다. 다음 질문에 답하시오.
① 메모리 참조가 200나노초 걸린다면, 페이지로 된 기억장치의 참조는 얼마나 걸리는가?
400나노초 : (페이지 테이블 액세스 시간 = 200 + 메모리 액세스 시간 = 200)
페이지 테이블에 액세스하기 위해 200나노초와 메모리 단어를 액세스하기 위해 200나노초.

② 연관 레지스터를 추가하여 모든 페이지 테이블 참조의 75%를 연관 레지스터에서 찾는다면, 실제 메모리 접근시간은 얼마나 되는가? (단, 연관 레지스터에서 어떤 페이지 테이블 항목을 찾을 때 해당 항목이 그곳에 있다면 시간은 걸리지 않는다고 가정한다).
0.75 X (200나노초) + 0.25 X (400나노초) = 250나노초

59. 페이지 테이블에서 항목 2개가 메모리의 동일한 페이지 프레임을 가리킨다면 그 결과는 어떻게 될까? 많은 양의 메모리를 한 장소에서 다른 장소로 복사할 때 이것을 사용하여 어떻게 필요한 시간의 양을 감소시킬 수 있는가? 한 페이지를 다른 페이지로 업데이트했을 때 어떤 효과가 있는가?
하나의 페이지에 있는 두 개의 항목들을 메모리에서 동일한 페이지로 나타나게 하면 사용자들은 코드와 데이터를 공유할 수 있다. 만일 그 코드가 재진입한다면 많은 메모리공간은 text editors, compilers, database systems등과 같은 큰 프로그램들을 나눠서 사용하므로 저장 수 있다. 많은 양의 메모리를 복사하는 것은 서로 다른 페이지 테이블들을 동일한 메모리 할당을 나타내게 하므로 효과가 나타날 수 있다. 그렇지만 재진입이 불가한 코드나 데이터들을 공유하는 것은 그 코드를 사용할 수 있는 모든 사용자가 그것을 변경할 수 있다는 것과 이러한 변경들은 다른 사용자들이 복사를 할 수 있다는 것을 의미한다.
[참고]

다중 프로그램 컴퓨터 특히 시분할 시스템에서 여러 사용자가 동일한 프로그램을 수행하는 경우, 만약 각 사용자에게 개별적으로 이 프로그램의 복사본을 한 부씩 주어서 수행한다면 메인 메모리가 많이 낭비될 것이다. 따라서 메인 메모리의 낭비가 발생하는 것을 방지하기 위해 공유가 가능한 페이지를 사용한다. 공유는 한 프로세스 집단을 효과적으로 수행하는 데 필요한 메인 메모리의 양을 줄이는 동시에 주어진 시스템을 더 많은 사용자가 쓸 수 있도록 해준다.

60. 바인딩을 설명하시오.

하나의 프로그램에 각각 재적재할 수 없는 주소에 의해 사용된 실제주소를 설정하는 것

61. 중첩(오버레이)의 개념과 단점을 설명하시오.

운영체제 영역과 메모리 공간의 일부 영역에 프로그램(작업) 실행에 반드시 필요한 명령어와 데이터를 적재시키고 나머지 영역, 즉 중첩 구동기(오버레이 드라이버) 영역에는 실행기간 동안에 필요한 각종 사용자 코드 등을 적재하여 필요한 시기에 해당되는 프로그램을 불러들여 수행하는 방법이다. 프로그래머들은 오버레이들이 서로 방해하지 않도록 프로그램과 데이터를 주의 깊게 설계해야 한다.

62. 동적 적재의 개념과 장점을 설명하시오.

루틴이 요청되었지만 아직 메모리상 존재하지 않을 때에만 루틴을 적재하는 것 그 루틴은 첫 번째로 요청되고 나서야 비로소 메모리로 적재될 수 있다
루틴들은 적재되기 전에는 결코 사용될 수 없다. 그러므로 더욱 자유로운 메모리이다.

63. 재할당 레지스터를 설명하시오.

하나의 프로그램을 위해 가장 낮은 물리 주소를 주기위해 사용되는 base 레지스터다.

64. 고정 분할 영역을 만드는 방법을 열거하시오.

- 기준(base) 레지스터와 한계(limit) 레지스터를 사용하는 방법
- 현재 실행중인 위/아래 바운드 레지스터를 사용하는 방법

65. 세그먼트를 설명하시오.

프로그램을 logical 세그먼트로 분리하고 이 세그먼트 공간을 메모리속으로 각각 할당하는 것 그 세그먼트들은 다양한 길이일수도 있고 연속해서 할당될 필요는 없다

66. 페이지 크기가 4KB이고 메모리 크기가 256KB인 메모리 페이징 시스템이 있다고 가정하여 다음 질문에 답하시오.

① 페이지 프레임 수는?

256KB/4KB = 64개

② 이 메모리 주소를 해결하는 데 필요한 비트는?

18비트

③ 페이지 번호에 사용하는 비트와 페이지 오프셋에 사용하는 비트는?
페이지 번호 = 6비트, 페이지 오프셋 = 12비트

67. 크기가 2^{12}바이트인 동일한 분할(영역)과 232바이트의 메인 메모리를 갖는 고정 분할 방법을 사용하는 시스템이 있다고 가정하자. 프로세스 테이블은 각 상주 프로세스의 분할 포인터를 유지한다. 프로세스 테이블의 포인터에 필요한 비트는?
프로세스 테이블 포인터는 시스템의 각 분할을 가리킬 수 있어야 한다. 예를 들어, 네 개의 분할이 있다면 우리는 각 파티션을 가리키도록 두 개의 비트가 필요하다(분할 A : 00_2, 분할 B : 01_2, 분할 C : (10_2), 분할 D : 11_2).
먼저 분할의 수를 계산한다.
분할 수 = (메인 메모리 크기) / (각 분할 크기) = 2^{32} / 2^{12} = 2^{20}.

따라서 2^{20}개의 분할을 가리키도록 정확히 20비트가 필요하다.

68 페이지 크기가 1KB라고 가정하면, 다음 주소 참조에서 페이지 번호와 오프셋은? (10진수로 제공한다.)
① 페이지 번호 = 1; 오프셋 = 327
② 페이지 번호 = 18; 오프셋 = 934
③ 페이지 번호 = 29; 오프셋 = 304
④ 페이지 번호 = 0; 오프셋 = 256
⑤ 페이지 번호 = 16; 오프셋 = 1

논리주소 (십진수)	논리주소(2진수)	페이지 수 (6비트,2진수)	오프셋 (10 비트,2진수)	페이지 수 (십진수)	오프셋 (십진수)
2375	0000 1001 0100 0111	0000 10	01 0100 0111	2	327
19366	0100 1011 1010 0110	0100 10	11 1010 0110	18	934
30000	0111 0101 0011 0000	0111 01	01 0011 0000	29	304
256	0000 0001 0000 0000	0000 00	01 0000 0000	0	256
16385	0100 0000 0000 0001	0100 00	00 0000 0001	16	1

69. 페이징 시스템에서 프로세스는 소유하지 않는(적재되지 않는) 메모리에 액세스할 수 없다. 다음 질문에 답하시오.
① 프로세스가 소유하지 않는(적재되지 않는) 메모리에 액세스할 수 없는 이유는?
페이징 시스템의 주소는 논리적 페이지 번호와 오프셋이다. 물리적 페이지는 물리적 페이지 번호는 물리적 페이지를 생성하는 논리 페이지 번호에 기반하는 테이블을 검색함으로써 발견된다. 운영 체제는 이 테이블의 내용을 제어하기 때문에, 오직 그 프로세스에 할당된 물리 페이지에 액세스하도록 프로세스를 제한할 수 있다. 따라서 프로세스는 그 페이지가 페이지 테이블에 없기 때문에 페이지를 참조할 방법이 없다.

② 운영체제가 다른 메모리에 액세스할 수 있을까? 그리고 운영체제가 다른 메모리에 액세

스할 수 있게 해야 하는가, 아니면 하지 않아도 되는가?

이러한 액세스를 허용하기 위해, 운영 체제는 비 프로세스(적재가 안된) 메모리(프레임)에 대한 항목은 프로세스의 페이지 테이블에 단순하게 추가되도록 할 필요가 있다. 두 개 이상의 프로세스가 다른 논리 주소라도 동일한 물리적 주소로 데이터 교환 즉, 읽고(read), 작성(write)해야 할 때 유용하다. 이것은 매우 효율적인 프로세스간 통신이며 또한 공유 페이지 구현에도 유용하다.

70. 페이징 시스템에서 페이지 테이블이 어떤 기능을 하는지 설명하시오.

특정 프로세스의 페이지 테이블은 메모리의 특정 페이지 번호에 대응하는 물리적 메모리에 있는 페이지(프레임)의 번호를 포함한다. 논리 주소를 물리 주소로 변환하는 메모리 관리 장치 (MMU)로 사용된다.

71. 컴파일러에서는 함수 호출이나 순환의 시작 부분으로 점프하는 코드와 데이터를 참조하는 주소를 생성해야 한다. 메모리 시스템에서 이 주소는 물리적 주소인가, 아니면 논리적 주소인가?

논리적 주소

Chapter 8 가상 메모리_연습문제

1	②	**16**	④	**31**	①	**46**	②
2	④	**17**	②	**32**	③	**47**	③
3	①	**18**	④	**33**	③	**48**	①
4	④	**19**	④	**34**	①	**49**	④
5	②	**20**	①	**35**	③	**50**	①
6	②	**21**	①	**36**	④	**51**	①
7	③	**22**	④	**37**	④	**52**	④
8	③	**23**	③	**38**	③	**53**	②
9	①	**24**	③	**39**	④	**54**	④
10	③	**25**	④	**40**	④	**55**	②
11	①	**26**	②	**41**	①	**56**	②
12	①	**27**	①	**42**	③	**57**	②
13	①	**28**	④	**43**	①	**58**	④
14	①	**29**	①	**44**	①	**59**	①
15	④	**30**	④	**45**	①	**60**	②

61. 요구 페이징과 프리 페이징을 설명하시오.

- 요구 페이징 : 프로세스 실행시에 필요한 페이지가 메인 메모리에 없을 때 즉 페이지 부재가 발생되면 디스크로부터 필요한 페이지를 읽어 온다.
- 프리 페이징 : 디스크에 한번 접근 할 때 연속된 페이지들이 참조될 것으로 예상하여 요구가 없더라도 미리 인접한 페이지들을 메인 메모리에 읽어 들이는 것이다.

62. 요구 페이지의 장점을 열거하시오.

빈 물리적 메모리와 스왑시간을 줄이고 다중프로그래밍의 정도를 높은 수준까지 이끈다.

63. 페이지 부재란 무엇인지 설명하시오.

아직은 메모리에 없는 하나의 특정 페이지를 필요로 하는 프로그램에 의해 인터럽트로 나타난다. 즉 저장되지 않은 페이지를 사용하려고 한다면 페이지 부재(Page Fault)가 발생된다.

64. 페이지 부재를 수행하는 여섯 단계를 열거하시오.

① 프로세스 제어 블록에 있는 내부 테이블을 검사하여 프로세스 참조가 메모리 액세스에 타당한지, 부당한지를 결정한다.
② 만약, 프로세스 참조가 무효화되었으면 프로세스는 중단된다. 프로세스가 유효한 참조로 페이지를 아직 가져오지 않았다면 페이지를 가져와야 한다.
③ 비어있는 프레임 리스트 중 하나를 선택한다.

④ 할당된 프레임에 요구된 페이지를 읽어 들이기 위해 디스크 동작을 스케줄한다.
⑤ 요구된 페이지가 메모리에 있다는 것을 알리기 위해 페이지 테이블을 수정한다.
⑥ 주소 트랩에 의해 인터럽트된 명령어들을 다시 시작한다.

65. 페이지 대치란 무엇인지 설명하시오.

선호되지 않는 어떤 프레임(희생자)을 버리는 것(스왑 아웃)으로 물리적 메모리에 비어있는 프레임이 없다는 것을 발견하게 되면(페이지 부재가 발생하면) 메인 메모리에 있으면서 사용되지 않는 페이지를 없애고 새로운 페이지로 바꾸는 과정을 의미

66. 스래싱이란 무엇인지 설명하시오.

계속적으로 페이지 교환이 일어나는 현상을 '스래싱(thrashing)'이라 한다. 만약, 어떤 프로세스가 프로세스 수행에 보내는 시간보다 페이지 교환에 보내는 시간이 더 크면 "스래싱을 하고 있다"고 말한다. 스래싱은 어떤 프로그램을 수행하는 데 있어서 최소 페이지수를 적게 할당할 때 일어나는데 계속해서 페이지 부재를 만들어 낸다.

67. 작업 집합 모델의 장점을 설명하시오.

각각의 프로세스에 대한 최적에 가까운 크기로 결정을 가능하게 하고 스레싱을 예방하는 프레임 할당과 다중프로그래밍의 높은 정도(수준)를 가능하게 한다.

68. 단순 페이지 시스템을 다음 매개변수 관점에서 살펴보자.

① 논리적 주소에서는 몇 개의 비트가 있는가?

논리적 메모리 주소 영역의 바이트 수는 2^{16}페이지이면서 페이지 크기는 2^{10} 이므로 전체 논리적 메모리 주소 영역은 226바이트로 26 비트의 논리적 주소가 필요하다

② 프레임에 몇 개의 바이트가 있는가?

하나의 프레임은 하나의 페이지 2^{10}바이트와 동일한 크기이다

③ 물리적 주소에서 프레임을 나타내는 비트는 몇 개인가?

메인 메모리속의 프레임 수는 (2^{32}bytes of main memory)/(2^{10}bytes/frame) = 2^{22} 프레임 즉 프레임을 표시하려면 22비트가 필요하다.

④ 페이지 테이블에 있는 항목들은 몇 개인가?

논리적 주소에는 각각의 페이지에 해당하는 하나의 항목이 있다 그러므로 216 항목들이 있다.

⑤ 각 페이지 테이블 항목에는 몇 개의 비트가 있는가? (단, 각 페이지 테이블 항목은 유효·무효 비트를 포함하고 있다고 가정한다.)

유효 /무효비트를 추가하면 전체 23비트의 경우 22비트가 메인 메모리의 프레임 위치를 명확히 하기 위해서 필요하다.

69. 프리 페이징을 사용하는 이유를 설명하시오.

각각의 프로세스는 초기화를 수행하기 위해 최소 페이지를 요구한다. 나중에 페이지를 줄이기 위해 그것들을 즉시 가져온다.

70. 페이지 크기를 결정하는 요소를 열거하시오.
- 한 개의 큰 페이지는 페이지테이블 크기를 작게 하는 것이 가능하다.
- 내부 단편화를 축소시키기 위해서는 페이지 크기를 줄여야만 한다.
- 입출력 시간을 줄이기 위해서는 페이지 사이즈를 크게 해야 한다.
- 지역성이 향상됨에 따라, 입출력의 양은 조금 더 작은 페이지를 위해 감소된다.
- 페이지 부재수를 줄이기 위해서는 큰 페이지가 필요하다.

71. 시스템 하나에 32비트 가상 주소와 24비트 물리적 주소가 있을 때, 페이지 하나가 16KB라면 페이지 테이블에는 항목이 몇 개 있어야 하는가?
$2^{32}/2^{14}$(16KB) = 2^{18}(262,144)

72. 역 페이지 테이블과 종래의 페이지 테이블을 비교 설명하시오.
종래의 페이지 테이블에서는 가상주소 공간에서 페이지별로 나누었을 때 가상 페이지 개수만큼의 항목이 필요하다. 이때 페이지 테이블의 색인 번호는 가상 주소번호이며 각 페이지 테이블안에는 물리 페이지 번호들이 들어가 있다. 이럴 경우 페이지 테이블의 크기가 커지게 되므로 다른 대안으로 나온게 역 페이지 테이블이다. 물리메모리에서 페이지별로 나누었을 때 물리 페이지 개수만큼의 항목이 필요한데 이 물리 페이지 번호가 역 페이지 테이블에서는 색인 번호로 사용되며 역 페이지 테이블 안에는 그 물리 페이지에 사상된 가상 페이지 번호가 들어 있다. 역 페이지 테이블 의 경우페이지 테이블을 줄일 수 있다는 장점이 있는 반면에 가상 주소를 물리 주소로 변환하는 과정이 복잡하다.

73. 48비트 가상 주소와 32비트 물리적 주소를 가진 기계가 있다. 페이지는 8KB이다. 페이지 테이블은 항목 몇 개로 구성되는가?
$2^{48}/2^{13}$ = 2^{35} = 2^{5} * 2^{30} = 32G 항목

74. 32비트(4GB) 가상 주소와 1GB 물리적 주소, 1MB 페이지를 사용하는 시스템이 있다. 다음 물음에 답하시오.
① 주소 공간에 가상 페이지가 몇 개 있는가? 4096
② 주소 공간에 물리적 페이지가 몇 개 있는가? 1024
③ 오프셋에 얼마큼 비트가 있는가? 20
④ 가상 페이지 번호에 비트가 몇 개 필요한가? 12
⑤ 물리적 페이지 번호에 비트가 몇 개 필요한가? 10

75. 다음 페이지 참조열에서 ①~③의 대치 알고리즘에 페이지 부재가 몇 개 일어나는가? (단, 페이지 프레임은 4개로 가정한다.)
① LRU
② 선입선출FIFO
③ 최적optimal

LRU - 9회

0	6	5	1	4	5	0	5	3	6	5	5	2	3	2	3	5	2	2	6
0	0	0	0	4	4	4	4	4	6	6	6	6	6	6	6	6	6	6	6
	6	6	6	6	6	0	0	0	0	0	0	2	2	2	2	2	2	2	2
		5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
			1	1	1	1	1	3	3	3	3	3	3	3	3	3	3	3	3

선입선출(FIFO) - 10회

0	6	5	1	4	5	0	5	3	6	5	5	2	3	2	3	5	2	2	6
0	0	0	0	4	4	4	4	4	4	5	5	5	5	5	5	5	5	5	5
	6	6	6	6	6	0	0	0	0	0	0	2	2	2	2	2	2	2	2
		5	5	5	5	5	5	3	3	3	3	3	3	3	3	3	3	3	3
			1	1	1	1	1	1	6	6	6	6	6	6	6	6	6	6	6

최적(optimal) - 7회

0	6	5	1	4	5	0	5	3	6	5	5	2	3	2	3	5	2	2	6
0	0	0	0	0	0	0	0	3	3	3	3	3	3	3	3	3	3	3	3
	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
		5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
			1	4	4	4	4	4	4	4	4	2	2	2	2	2	2	2	2

Chapter 9 입출력 시스템과 디스크 관리_연습문제

1	④	**11**	④	**21**	②	**31**	③
2	①	**12**	③	**22**	③	**32**	③
3	②	**13**	①	**23**	③	**33**	④
4	②	**14**	②	**24**	②	**34**	①
5	②	**15**	②	**25**	③	**35**	③
6	④	**16**	④	**26**	④	**36**	②
7	④	**17**	③	**27**	②	**37**	②
8	②	**18**	③	**28**	③	**38**	④
9	②	**19**	①	**29**	②		
10	④	**20**	③	**30**	①		

39. 입출력 모듈의 기능을 간략하게 설명하시오.

- 주변(입출력) 장치의 제어
- 프로세서와의 통신
- 데이터 버퍼링
- 오류검출

40. DMA를 설명하시오.

프로세서의 도움 없이 메인메모리로부터 직접적으로 제어하여 데이터 전송이 이루어지는 형태로 프로세서는 시작 주소와 방향 그리고 길이를 DMA에게 전달해주면 DMA는 직접 작업을 처리하게 된다.

41. DMA 방법은 어떻게 병렬 처리 시스템의 성능을 향상시키는가? 하드웨어 설계가 복잡해지는 이유는?

DMA 기법은 DMA 시스템이 데이터를 시스템과 메모리 버스들을 거쳐 전이되는 동안 프로세서가 작업을 수행하게 하여 시스템의 병렬성(동시성)을 증가시킨다. 하드웨어를 디자인 하는 것은 DMA 제어기가 그 시스템으로 통합되어야만 하고 그 통합된 시스템은 DMA 제어기가 버스관리자(bus master)가 되어야 하기 때문에 복잡하다.

42. 디스크 액세스 시간(속도)을 결정하는 요소는?

이동 디스크의 데이터 액세스 시간 : 탐색 시간 + 회전 지연 시간 + 전송 시간
고정 헤드의 디스크 시스템 : 회전 지연 시간 + 전송 시간

43. 디스크 입출력에 필요한 정보는?

- 입력 동작인가 아니면 출력 동작인가
- 디스크 주소(구동기, 실린더, 표면, 블록)

- 메모리 주소
- 전송할 정보의 총량(바이트 또는 단어의 수)

44. 프로세서 속도를 높일 때 시스템 버스와 장치 속도 크기를 증가시켜야 하는 이유는? 입출력 50%와 연산(프로세서 작업) 50%를 수행하는 하나의 시스템을 고려하여 설명하시오.

이 시스템에서 프로세서 수행을 더블링(doubling)하는 것은 전체 시스템 수행을 단지 50%까지 증가시킬 것이다. 두 개의 시스템측면들을 더블링하면 수행을 100%까지 증가시킬 것이다. 일반적으로, 개별적인 시스템 구성요소들의 수행을 증가시키는 것보다 최근 시스템 병목현상을 제거하고, 전체시스템 수행을 증가시키는 것이 중요하다.

45. 디스크 스케줄링 방법 중 최소 탐색 시간 우선 스케줄링을 설명하시오.

디스크 요청을 처리하기 위해서 헤드가 먼 곳까지 이동하기 전에 현재 헤드 위치에 가까운 모든 요구를 먼저 처리하는 방법이다. 최소 위치결정 시간 우선(SPTF, Shortest-Positioning-Time-First) 알고리즘이라고도 한다.

46. 최소 탐색 시간 우선 스케줄링은 실린더의 너무 안쪽이나 바깥쪽보다 중간쯤이 좋다. 그 이유를 설명하시오.

디스크의 중심은 모든 다른 트랙들에 대해 가장 작은 평균적인 거리에 위치한다. 그러므로 그 알고리즘이 첫 번째 요청을 서비스 한 후에는 또 다른 특정한 트랙보다 좀 더 중심트랙에 가까워지며 그 결과 그 알고리즘은 처음 그 위치로 더 자주 가게 된다. 일단 하나의 특정한 트랙에서 최소 탐색시간 우선(SSTF)은 이 트랙 가까이 중심부에 유지되는 경향이 있으며 그 결과 이미 스캐줄된 전략은 초기형태를 중심부에 만들 것이다

47. 선입선처리 스케줄링의 문제점은?

요청이 도착한 순서에 따라 처리하므로 헤드는 가까이 사용될 필요한 몇몇의 트랙들을 일시적으로 생략하면서 광범위한 트랙사이에서 전체적으로 움직이므로 매우 비효율적이다.

48. 단일 사용자 환경에서 선입선처리 외의 다른 스케줄링이 유용한가?

단일 사용자한경에서, 그 I/O 대기행렬(queue)은 대개 길이가 1이다. 그러므로 선입선처리(FCFS)는 디스크를 스캐줄링하는 가장 경제적인 방법이다.

49. C-SCAN 방법은 SCAN 방법을 어떻게 변경한 것인가?

C_SCAN은 가장 높은 트랙수에 도달 한 후에는, 어떤 경로도 거치지 않고 가장 낮은 트랙수를 요청하도록 변경된다. 대기 시간을 좀더 균등하게 하려고 스캔 알고리즘을 변형시킨 알고리즘이다. 스캔(SCAN) 스케줄링과 같이 한쪽 방향으로 헤드를 이동해 가면서 요청을 처리하지만 한쪽 끝에 다다르면 반대 방향으로 헤드를 이동하는 것이 아니라 다시 처음부터 처리를 한다. 따라서 C-SCAN은 처음과 마지막 트랙을 서로 인접시킨 것과 같은 원형처럼 디스크를 처리하므로 처리량을 향상시키면서 바깥 트랙과 안쪽 트랙에 대한 차별이 없어 반응시간의 변화를 줄이는 효과를 준다.

50. 다음과 같은 선형 요청 디스크 큐가 있다. ①~③의 디스크 스케줄링 알고리즘에서 트랙의 헤드 이동 수는?

① 393 = 15 + 2 + 44 + 37 + 20 + 30 + 11 + 32 + 52 + 50

② 126 = 15 + 2 + 7 + 10 + 11 + 16 + 4 + 50 + 2 + 9

③ 175 = 15 + 5 + 37 + 16 + 11 + 17 + 4 + 9 + 11 + 50

51. RAID를 사용할 때의 장점, RAID의 0~3계층까지 구조와 차이점을 비교 설명하시오.

계속적으로 증대된 프로세서 성능에 비해 디스크 성능은 상대적으로 별로 증대되지 못했다. 이를 개선하기 위해 디스크 입출력시 병렬처리를 적용하게 되었는데 이것이 RAID 이며 여러 대의 물리적 디스크를 논리적으로는 하나의 하드 디스크로 인식하는 기술이다.

- RAID 0: strip(data group) 단위로 데이터를 여러 개의 디스크 드라이브에 분산하는 구조, 여러 개의 디스크 드라이브를 동시에 액세스함으로써 성능 개선이 가능함
- RAID 1: RAID 0와 같은 방식으로 각 디스크 드라이브에 대해 별도의 백업 드라이브 운영
- RAID 2: RAID 0, 1은 스트립 단위로 운영되는 반면에 RAID 2, 3는 워드 단위로 운영 워드를 니브(4비트) 단위로 나누어 각 비트별로 다른 디스크 드라이브에 저장되며, 그 비트에 대해 해밍 코드 3비트를 생성하여 이 코드도 각기 다른 디스크 드라이브에 1비트씩을 배치한다. 하나의 니블을 표현하기 위해 총 7비트가 사용된다.
- RAID 3: RAID 2의 간소화된 버전. 3비트의 해밍 코드 대신 1비트의 패리티 비트를 사용한다. 하나의 니블을 표현하기 위해 총 5비트가 사용된다.

Chapter 10 파일 관리_연습문제

1	①	**16**	④	**31**	①	**46**	③
2	③	**17**	①	**32**	②	**47**	④
3	②	**18**	①	**33**	②		
4	③	**19**	④	**34**	②		
5	④	**20**	④	**35**	④		
6	③	**21**	④	**36**	③		
7	②	**22**	①	**37**	①		
8	④	**23**	①	**38**	④		
9	②	**24**	④	**39**	①		
10	①	**25**	③	**40**	④		
11	①	**26**	③	**41**	①		
12	④	**27**	③	**42**	③		
13	①	**28**	①	**43**	④		
14	④	**29**	④	**44**	③		
15	④	**30**	③	**45**	④		

48. 메인 메모리보다 대용량의 기억장치에서 파일을 할당하는 비트맵을 사용해야 하는 이유는?

비트 맵이 메인 메모리에 저장되면 시스템 고장(메모리 오류)인 경우라도 빈 공간 리스트가 손실 되지 않는다.

49. 연속 할당 정책, 연결 할당 정책, 인덱스 할당 정책을 지원하는 시스템이 있다. 할당 정책을 선택하는 기준과 특정 파일에서는 어떤 할당 정책을 이용해야 하는지 설명하시오.

- 연속 : 파일이 비교적 작은 경우, 파일은 일반적으로 순차적으로 액세스된다.
- 연결 : 파일이 크고 일반적으로 순차적으로 액세스되는 경우.
- 인덱스 : 파일이 크고 일반적으로 직접 액세스되는 경우.

50. 파일의 액세스 속도에서 할당 방법을 평가한다면?

연속 할당이 가장 빠르다. 디스크 헤드가 파일의 액세스 사이를 이동해야 할 수도 있기 때문에 링크는 느립니다. 전체 인덱스가 항상 메모리에 보존 할 수없는 경우 인덱스 할당은 최저 이다. 그렇지 않다면, 파일 인덱스의 다음 블록을 액세스 할 때 추가 시간이 필요하다.

51. 각 디스크 블록이 8KB(213바이트)인 128GB(237바이트) 디스크의 파일 시스템이 있는 컴퓨터를 가정하자. 이 컴퓨터의 운영체제가 FAT를 사용할 때 FAT가 사용하는 최소 메모리 크기는?

각 디스크 블록에 FAT 항목이 있으므로 디스크는 2^{37}바이트, 디스크 블록은 2^{13}바이트이다. 따라서 디스크 블록의 수 즉 FAT 항목 수는 $2^{37}/2^{13} = 2^{24}$이다. 그러므로 2^{24} 항목에 대한 블록 번호(디스크 주소)는 $\log(2^{24}) = 24$비트 즉 최소 3바이트를 필요로 한다. FAT에 의해 점유 된 공간의 최소 크기는 항목당 3바이트이므로 2^{24} X 3 = 16MB X 3 = 48MB

52. 파일을 포함한 디스크 블록을 구성하는 많은 방법이 있다. 다음 물음에 답하시오.

① 연속 할당이 인덱스 할당보다 어떤 장점이 있는지 설명하시오.

이 구현은 인덱스 구조의 복잡한 대신 첫 번째 블록과 길이에 대한 포인터만 저장하므로 단순하다. 이 단순함은 이러한 데이터 구조의 디스크 공간을 적게 사용하고 작은 번호 탐색은 즉, 다음 블록 에 대한 포인터를 얻기 위해 다시 인덱스 계속 할 필요가 없으므로 파일 액세스에 빠르다.

② 인덱스 할당이 연속 할당보다 어떤 장점이 있는지 설명하시오.

인덱스 할당은 큰 연속 공간에 대한 필요성이 없기 때문에 파일 공간 확장이 쉽고 연속 할당 가능하고 외부 단편화를 방지할 수 있다. 인덱스 할당은 연속 할당 보다 직접 액세스에 더 효율적이다. 다중 인덱스 할당(Multilevel Indexed Allocation)은 포인터가 파일 디스크립터가 아닌 별도의 인덱스 블록에 저장 될 수 있기 때문에 작은 파일 더 효율적이다.

53. 디스크 크기가 40GB라고 하자. 파일에서 다음 블록의 블록 번호 4바이트가 필요하고, 기타 데이터 2바이트가 필요하다면 각 블록의 크기가 1K일 때 FAT 테이블의 크기는?

디스크 블록 수 = FAT 의 항목 수 = 40GB/1KB = 40 X 1024 X 1024

따라서 FAT 크기는 6 X 40 X1024 X1014 = 240MB

Chapter 11 분산 및 다중(병렬) 처리 시스템_연습문제

1	②	**16**	④	**31**	④	**46**	②
2	①	**17**	②	**32**	②	**47**	①
3	②	**18**	②	**33**	④	**48**	②
4	①	**19**	③	**34**	④	**49**	②
5	①	**20**	③	**35**	②	**50**	②
6	②	**21**	①	**36**	③	**51**	②
7	③	**22**	③	**37**	②	**52**	③
8	②	**23**	②	**38**	②	**53**	①
9	①	**24**	④	**39**	①	**54**	④
10	①	**25**	③	**40**	②	**55**	①
11	③	**26**	④	**41**	④	**56**	①
12	①	**27**	④	**42**	②	**57**	④
13	③	**28**	④	**43**	②	**58**	②
14	①	**29**	①	**44**	④	**59**	②
15	②	**30**	③	**45**	②	**60**	②
						61	②

62. 분산 시스템의 장점은?

분산 시스템은 단일 시스템보다 적은 비용으로 여러 서버에서 사용자 요청을 동시에 처리하여 높은 성능을 얻어낼 수 있다. 또한 많은 사용자가 동일한 파일에 효과적이고 높은 신뢰성을 가지고 접근할 수 있게 해준다.

63. 분산 시스템은 왜 미들웨어를 요구하는가?

분산시스템은 서로 다른 운영체제와 서버 프로그램과의 호환성뿐만 아니라 서로 다른 통신 프로토콜을 사용하는 네트워크간의 접속, 네트워크 자원에 대한 액세스 그리고 시스템을 연결해 단일한 사용자 환경으로 만들어 이기종 머신들이 한 컴퓨터처럼 함께 동작할 수 있도록 클라이언트/서버 사이에서 교량적인 역할을 하는 소프트웨어 서비스를 의미한다.

64. 주종 다중 처리기 운영체제 시스템의 구조와 특징을 설명하시오.

하나의 주(M) 프로세서와 나머지 종(S) 프로세서로 구성된다. 주(M) 프로세서는 입출력과 연산을 수행하고, 종(S) 프로세서는 연산만을 수행한다. 종(S) 프로세서에서 입출력 발생시 주(M) 프로세서에게 서비스를 요청한다. 주(M) 프로세서의 고장시 전 시스템이 멈춘다. 주(M) 프로세서만이 운영 체제를 수행한다.

65. 다중 프로세서 운영체제의 분리 실행을 설명하시오.

다중 프로세서 운영 체제의 구성 방법 중의 하나로서 다음과 같은 특징이 있다.

- 각 프로세서가 서로 다른 운영 체제를 가지고 있으며, 각 프로세서에서 발생하는 인터럽트도 해당 프로세서에서 독립적으로 해결한다.
- 각 프로세서는 자신만의 파일과 입출력 장치를 제어한다.
- 각 프로세서는 서로 수행을 서로 돕지 않아 어떤 유휴 상태이고 다른 프로세서는 바쁠 수도 있다.
- 한 프로세서의 고장으로 전 시스템이 멈추지 않는다.

66. 분산 시스템에서 요구되는 네 가지 유형의 투명성을 정의하고 각각의 예를 기술하시오.

- 위치 투명성 : 자원 위치와 각 컴포넌트가 상호작용하는 위치를 사용자가 몰라도 된다. 따라서 사용자는 지역 파일에 액세스하듯 원격 파일에 액세스하여 어떤 서버가 해당 파일을 보유하는지 알지 못한다. 예 : 웹페이지, NFS(네트워크 파일 시스템)
- 고장 투명성 : 시스템 구성 요소(컴포넌트)와 통신 오류 때문에 시스템을 수행하는 데 장애를 받지 않게 한다. 여러 자원이나 컴퓨터에 오류가 발생할 때 시스템 사용자는 성능이 떨어지는 정도만 느낄 수 있다. 오류를 시스템에서 제거하고 재사용할 수 있도록 회복시켜 준다. 대체로 복제나 복구를 이용하여 구현한다. 복제한 자원 중 하나만 남고 모두 고장이 나더라도 분산 시스템은 기능을 계속할 수 있다. 예 : 데이터베이스 관리 시스템
- 중복 투명성 : 시스템에 자원 사본이 여러 개 있다는 사실을 감춘다. 동일한 자원이 다수의 컴퓨터에 있더라도 사용자에게는 자원 하나로만 보이게 한다. 즉, 복제한 자원 그룹에서 모든 액세스가 자원이 하나만 있는 것처럼 보이게 하여 신뢰성과 유용성을 높일 수 있다. 예 : 분산 DBMS, 웹페이지 미러링
- 이동(이주) 투명성 : 자원을 한 시스템에서 다른 시스템으로 이동해도 사용자가 이를 의식하지 않고 이용할 수 있도록 하는 것이다. 예를 들어, 파일을 한 서버에서 다른 서버로 이동하는 것처럼 한 객체를 한 위치에서 다른 위치로 변경할 수 있게 한다. 예 : 웹페이지, NFS
- 영속 투명성 : 자원이 저장된 위치(메모리나 디스크) 정보를 감춘다.
- 자원 투명성 : 구성 요소에서 자원의 배당과 해제 정보를 감춘다. 자원을 공유하는 데 제공된다.
- 트랜잭션투명성 : 공유 공간에서 동작하는 트랜잭션 연산 조정과 자원 집합 사이의 결합을 숨겨 데이터 무결성과 일관성을 확보할 수 있게 한다.
- 재배치 투명성 : 한 객체의 재배치를 이와 통신하는 다른 객체에 감출 수 있게 한다.
- 규모 투명성 : 구성 요소를 추가하거나 제거하는 등 규모가 바뀌어도 사용자가 의식하지 않는다. 예 : 웹페이지
- 병행 투명성 : 사용자와 응용 프로그램이 서로 간섭 없이 공유 데이터 또는 객체에 동시에 액세스할 수 있다. 분산 시스템에서 매우 복잡한 메커니즘이 필요하다. 예 : NFS, 금융 자동화 기기 네트워크

Chapter 12 시스템 보안과 운영체제_연습문제

1	②	**11**	②	**21**	①
2	①	**12**	②	**22**	②
3	③	**13**	②		
4	③	**14**	③		
5	①	**15**	②		
6	③	**16**	③		
7	④	**17**	④		
8	②	**18**	④		
9	①	**19**	①		
10	④	**20**	③		

Chapter 13 유닉스 운영체제_연습문제

1	②	21	②	41	②	61	②
2	④	22	①	42	②	62	③
3	③	23	④	43	②	63	③
4	②	24	②	44	①	64	④
5	④	25	②	45	④	65	③
6	④	26	②	46	①	66	①
7	②	27	②	47	④	67	②
8	③	28	①	48	②	68	①
9	①	29	③	49	②	69	④
10	②	30	③	50	①	70	④
11	③	31	③	51	②	71	①
12	③	32	③	52	③		
13	④	33	③	53	②		
14	④	34	①	54	④		
15	②	35	①	55	③		
16	②	36	③	56	①		
17	③	37	②	57	③		
18	②	38	②	58	①		
19	②	39	①	59	③		
20	②	40	①	60	③		