4 과목 | 운영체제

20.8, 20.6, 19.8, 19.4, 19.3, 18.8, 18.4, 17.8, 17.5, 17.3, 16.8, 16.5, 16.3, 15.8, 15.5, 15.3, 14.8, 14.5, 14.3, …

핵심

150 운영체제의 개요

정의	컴퓨터 시스템의 자원들을 효율적으로 관리하며, 사용자가 컴퓨터를 편리하고 효과적으로 사용할 수 있도록 환경을 제공하는 여러 프로그램의 모임으로, 제어 프로그램과 처리 프로그램으로 구분 • 제어 프로그램 : 시스템 전체의 작동 상태 감시, 작업의 순서 지정, 작업에 사용되는 데이터 관리 등의 역할 수행 – 감시 프로그램(Supervisor Program) – 작업 제어 프로그램(Job Control Program) – 자료 관리 프로그램(Data Management Program) • 처리 프로그램 : 제어 프로그램의 지시를 받아 사용자가 요구한 문제를 처리하기 위한 프로그램 – 언어 번역 프로그램(Language Translator Program) – 서비스 프로그램(Service Program) – 문제 프로그램(Problem Program) • 운영체제의 종류 : Windows, UNIX, LINUX, MS-DOS 등
목적 및 성능 평가 기준	• 처리 능력 및 신뢰도 향상, 사용 가능도 향상, 반환 시간 의 단축 • 성능 평가 기준 – 처리 능력(Throughput) : 일정 시간 내에 시스템이 처리하는 일의 양 – 반환 시간(Turn Around Time) : 시스템에 작업을 의뢰한 시간부터 처리가 완료될 때까지 걸린 시간 – 사용 가능도(Availability) : 시스템의 각종 자원을 사용할 필요가 있을 때 즉시 사용 가능한 정도 – 신뢰도(Reliability) : 시스템이 주어진 문제를 정확하게 해결하는 정도
기능	• 프로세서, 기억장치, 입·출력장치, 파일 및 정보 등의 자원 관리 • 자원의 스케줄링 기능 제공 • 사용자와 시스템 간의 편리한 인터페이스 제공 • 시스템의 각종 하드웨어와 네트워크 관리·제어 • 시스템의 오류 검사 및 복구, 데이터 관리, 데이터 및 자원 공유, 시스템의 초기화 • 자원 보호 기능 제공 • 가상 계산기 기능 제공

20.8, 19.3, 18.4, 17.5, 17.3, 15.3, 14.8, 14.3, 12.5, 12.3, 10.9, 10.5, 09.8, 09.5, 08.9, 08.5, 08.3, 07.9, …

핵심

151 운영체제 운용 기법 및 발달 과정

운영체제 운용 기법

일괄 처리(Batch Processing) 시스템	• 초기의 컴퓨터 시스템에서 사용된 형태로, 일정량 또는 일정 기간 동안 데이터를 모아서 한꺼번에 처리하는 방식 • 컴퓨터 시스템을 효율적으로 사용할 수 있음

	• 사용자 측면에서는 반환(응답) 시간이 늦지만 하나의 작업이 모든 자원을 독점하므로 CPU 유휴 시간이 줄어듦 • 급여 계산, 지불 계산, 연말 결산 등의 업무에 사용
다중 프로그래밍(Multi-Programming) 시스템	• 하나의 CPU와 주기억장치를 이용하여 여러 개의 프로그램을 동시에 처리하는 방식 • 하나의 주기억장치에 2개 이상의 프로그램을 기억시켜 놓고, 하나의 CPU와 대화하면서 동시에 처리함
시분할(Time Sharing) 시스템	• 여러 명의 사용자가 사용하는 시스템에서 컴퓨터가 사용자들의 프로그램을 번갈아 가며 처리해 줌으로써 각 사용자에게 독립된 컴퓨터를 사용하는 느낌을 주는 것이며 라운드 로빈(Round Robin) 방식이라고도 함 • 여러 사용자가 각자의 단말 장치를 통하여 동시에 운영체제와 대화하면서 각자의 프로그램을 실행함 • 하나의 CPU는 같은 시점에서 여러 개의 작업을 동시에 수행할 수 없기 때문에, CPU의 전체 사용 시간을 작은 작업 시간량(Time Slice)으로 나누어서 그 시간량 동안만 번갈아 가면서 CPU를 할당하여 각 작업을 처리함 • 다중 프로그래밍 방식과 결합하여 모든 작업이 동시에 진행되는 것처럼 대화식 처리가 가능함
다중 처리(Multi-Processing) 시스템	• 여러 개의 CPU와 하나의 주기억장치를 이용하여 여러 개의 프로그램을 동시에 처리하는 방식 • 하나의 CPU가 고장나더라도 다른 CPU를 이용하여 업무를 처리할 수 있으므로 시스템의 신뢰성과 안정성이 높음
실시간 처리(Real Time Processing) 시스템	• 데이터 발생 즉시, 또는 데이터 처리 요구가 있는 즉시 처리하여 결과를 산출하는 방식 • 우주선 운행이나 레이더 추적기, 핵물리학 실험 및 데이터 수집, 전화교환장치의 제어, 은행의 온라인 업무, 좌석 예약 업무, 인공위성, 군함 등의 제어 업무 등 시간에 제한을 두고 수행되어야 하는 작업에 사용됨
다중 모드 처리(Multi-Mode Processing)	일괄 처리 시스템, 시분할 시스템, 다중 처리 시스템, 실시간 처리 시스템을 한 시스템에서 모두 제공하는 방식
분산 처리(Distributed Processing) 시스템	• 여러 개의 컴퓨터(프로세서)를 통신 회선으로 연결하여 하나의 작업을 처리하는 방식 • 각 단말 장치나 컴퓨터 시스템은 고유의 운영체제와 CPU, 메모리를 가지고 있음

발달 과정

일괄 처리 시스템 → 다중 프로그래밍, 다중 처리, 시분할, 실시간 처리 시스템 → 다중 모드 → 분산 처리 시스템

20.8, 17.8, 17.3, 14.5, 10.9, 06.9, 06.5, 05.9, 05.3, 03.8, 03.5, 02.3, 01.3, 00.5

핵심

152 링커/로더

링커

• 언어 번역 프로그램이 생성한 목적 프로그램들과 라이브러리, 또 다른 실행 프로그램(로드 모듈) 등을 연결하여 실행 가능한 로드 모듈을 만드는 시스템 소프트웨어로 Linkage Edit

or라고도 한다.

- 연결 기능만 수행하는 로더의 한 형태로, 링커에 의해 수행되는 작업을 링킹(Linking)이라 한다.

로더

정의	컴퓨터 내부로 정보를 들여오거나 로드 모듈을 디스크 등의 보조기억장치로부터 주기억장치에 적재하는 시스템 소프트웨어
기능	• 할당(Allocation) : 실행 프로그램을 실행시키기 위해 기억장치 내에 옮겨놓을 공간을 확보하는 기능 • 연결(Linking) : 부 프로그램 호출 시 그 부 프로그램이 할당된 기억장소의 시작주소를 호출한 부분에 등록하여 연결하는 기능 • 재배치(Relocation) : 디스크 등의 보조기억장치에 저장된 프로그램이 사용하는 각 주소들을 할당된 기억장소의 실제 주소로 배치시키는 기능 • 적재(Loading) : 실행 프로그램을 할당된 기억공간에 실제로 옮기는 기능
종류	• Compile And Go 로더 : 별도의 로더 없이 언어 번역 프로그램이 로더의 기능까지 수행하는 방식(할당, 재배치, 적재 작업을 모두 언어 번역 프로그램이 담당) • 절대 로더(Absolute Loader) : 목적 프로그램을 기억 장소에 적재시키는 기능만 수행하는 로더(할당 및 연결은 프로그래머가, 재배치는 언어 번역 프로그램이 담당) • 직접 연결 로더(Direct Linking Loader) : 일반적인 기능의 로더로, 로더의 기본 기능 4가지를 모두 수행하는 로더 • 동적 적재 로더(Dynamic Loding Loader) : 프로그램을 한꺼번에 적재하는 것이 아니라 실행 시 필요한 일부분만을 적재하는 로더

20.8, 19.4, 18.8, 18.3, 17.8, 17.5, 17.3, 16.8, 16.3, 15.8, 15.5, 15.3, 14.8, 14.5, 14.3, 13.8, 13.6, 13.3, 12.5, …

핵심

153 프로세스의 여러 가지 정의/PCB

프로세스의 여러 가지 정의

- 실행중인 프로그램
- PCB를 가진 프로그램
- 실기억장치(주기억장치)에 저장된 프로그램
- 프로세서가 할당되는 실체
- 프로시저가 활동중인 것
- 비동기적 행위를 일으키는 주체
- 지정된 결과를 얻기 위한 일련의 계통적 동작
- 목적 또는 결과에 따라 발생되는 사건들의 과정
- 운영체제가 관리하는 최소 실행 단위
- 프로세서 제어 블록의 존재로서 명시되는 것

PCB(Process Control Block)

- 운영체제가 프로세스에 대한 중요한 정보를 저장해 놓는 곳이다.

- 각 프로세스가 생성될 때마다 고유의 PCB가 생성되고 프로세스가 완료되면 PCB가 제거된다.
- PCB에 저장되어 있는 정보
  - 프로세스의 현재 상태
  - 포인터(부모 프로세스·자식 프로세스·프로세스가 위치한 메모리·할당된 자원에 대한 포인터)
  - 프로세스 고유 식별자
  - 스케줄링 및 프로세서의 우선순위
  - CPU 레지스터 정보
  - 주기억장치 관리 정보
  - 입·출력 상태 정보
  - 계정 정보

20.8, 18.8, 18.3, 15.5, 14.3, 12.8, 12.3, 11.8, 10.9, 10.3, 09.5, 09.3, 07.9, 06.3, 00.10

핵심

154 스레드(Thread)

- 스레드는 프로세스 내에서의 작업 단위이면서 시스템의 여러 자원을 할당받아 실행하는 프로그램의 단위이다.
- 하나의 프로세스에 하나의 스레드가 존재하는 경우에는 단일 스레드, 하나 이상의 스레드가 존재하는 경우에는 다중 스레드라고 한다.
- 프로세스의 일부 특성을 갖고 있기 때문에 경량(Light Weight) 프로세스라고도 한다.
- 자신만의 스택(Stack)과 레지스터(Register)를 가지며 독립된 제어 흐름을 갖는다.
- 프로세스의 구성을 제어의 흐름 부분과 실행 환경 부분으로 나눌 때, 프로세스의 실행 부분을 담당함으로써 실행의 기본 단위가 된다.
- 운영체제의 성능을 개선하려는 소프트웨어적 접근 방법이다.
- **스레드의 분류**

사용자 수준의 스레드	• 사용자가 만든 라이브러리를 사용하여 스레드를 운용함 • 속도는 빠르지만 구현이 어려움
커널 수준의 스레드	• 운영체제의 커널에 의해 스레드를 운용함 • 구현이 쉽지만 속도가 느림

- **스레드 사용의 장점**
  - 하나의 프로세스를 여러 개의 스레드로 생성하여 병행성을 증진시킬 수 있다.
  - 하드웨어, 운영체제의 성능과 응용 프로그램의 처리율을 향상시킬 수 있다.
  - 응용 프로그램의 응답 시간(Response Time)을 단축시킬 수 있다.
  - 실행 환경을 공유시켜 기억 장소의 낭비가 줄어든다.
  - 프로세스들 간의 통신이 향상된다.

- 스레드는 공통적으로 접근 가능한 기억장치를 통해 효율적으로 통신한다.

19.8, 19.4, 19.3, 18.4, 18.3, 14.8, 07.9, 06.5, 03.8, 02.9, 01.6, 00.7, 00.5

핵심

155 스케줄링

- **정의** : 프로세스가 생성되어 실행될 때 필요한 시스템의 여러 자원을 해당 프로세스에게 할당하는 작업
- **목적** : 공정성, 처리율 증가, CPU 이용률 증가, 우선순위 제도, 오버헤드 최소화, 응답 시간 최소화, 반환 시간 최소화, 대기 시간 최소화, 균형 있는 자원의 사용, 무한 연기 회피
- **성능 평가 기준** : 스케줄링의 목적 중 CPU 이용률, 처리율, 반환 시간, 대기 시간, 응답 시간은 여러 종류의 스케줄링 성능을 평가하는 기준이 됨
- **문맥 교환(Context Switching)** : 하나의 프로세스에서 다른 프로세스로 CPU가 할당되는 과정에서 발생되는 것으로, 새로운 프로세스에 CPU를 할당하기 위해 현재 CPU가 할당된 프로세스의 상태 정보를 저장하고, 새로운 프로세스의 상태 정보를 설정한 후 CPU를 할당하여 실행되도록 하는 작업

20.6, 18.4, 18.3, 17.5, 09.5, 08.9, 07.9, 07.5, 06.9, 06.5, 05.9, 05.5, 05.3, 04.5, 04.3, 03.8, 03.5, …

핵심

156 프로세서(스) 스케줄링의 종류

비선점(Non-preemptive) 스케줄링	• 이미 할당된 CPU를 다른 프로세스가 강제로 빼앗아 사용할 수 없는 스케줄링 기법 • 프로세스가 CPU를 할당받으면 해당 프로세스가 완료될 때까지만 CPU를 사용하므로 응답시간 예측이 용이함 • 모든 프로세스에 대한 요구를 공정하게 처리할 수 있음 • 일괄 처리 방식에 적합하며, 중요한(처리 시간이 짧은) 작업이 중요하지 않은(처리 시간이 긴) 작업을 기다리는 경우가 발생할 수 있음 • 종류 : FCFS(FIFO), SJF, 우선순위, HRN, 기한부
선점(Preemptive) 스케줄링	• 하나의 프로세스가 CPU를 할당받아 실행하고 있을 때 우선순위가 높은 다른 프로세스가 CPU를 강제로 빼앗아 사용할 수 있는 스케줄링 기법 • 우선순위가 높은 프로세스를 빠르게 처리할 수 있음 • 주로 빠른 응답 시간을 요구하는 대화식 시분할 시스템, 온라인 응용 등에 사용 • 종류 : SRT, 선점 우선순위, Round Robin, 다단계 큐(MQ), 다단계 피드백 큐(MFQ)

20.8, 20.6, 19.8, 19.4, 18.8, 18.4, 18.3, 17.8, 17.5, 17.3, 16.8, 16.5, 16.3, 15.8, 15.5, 15.3, 14.8, 14.5, 14.3, …

핵심

157 비선점 스케줄링의 종류

FCFS(First Come First Service)	• 준비상태 큐에 도착한 순서에 따라 차례로 CPU를 할당하는 기법 • 먼저 도착한 것이 먼저 처리되어 공평성은 유지 되지만 짧은 작업이 긴 작업을, 중요한 작업이 중요하지 않은 작업을 기다리게 됨
SJF(Shortest Job First)	• 실행 시간이 가장 짧은 프로세스에 먼저 CPU를 할당하는 기법 • 가장 적은 평균 대기 시간을 제공하는 최적 알고리즘
HRN(Hightest Response-ratio Next)	• 실행 시간이 긴 프로세스에 불리한 SJF 기법을 보완하기 위한 것으로, 대기 시간과 서비스(실행) 시간을 이용하는 기법 • 우선순위 계산 공식 = $\frac{\text{대기 시간} + \text{서비스 시간}}{\text{서비스 시간}}$
기한부(Deadline)	• 프로세스에게 일정한 시간을 주어 그 시간 안에 프로세스를 완료하도록 하는 기법 • 시스템은 프로세스에게 할당할 정확한 시간을 추정해야 하며, 이를 위해서 사용자는 시스템이 요구한 프로세스에 대한 정확한 정보를 제공해야 함
우선순위(Priority)	• 기다리는 각 프로세스마다 우선순위를 부여하여 그 중 가장 높은 프로세스에게 먼저 CPU를 할당하는 기법 • 우선순위의 등급은 내부적 요인과 외부적 요인에 따라 부여할 수 있음 • 각 작업마다 우선순위가 주어지며, 우선순위가 제일 높은 작업에게 먼저 프로세서가 할당됨 • 가장 낮은 순위를 부여받은 프로세스는 무한 연기 또는 기아 상태(Starvation)가 발생할 수 있음

잠깐만요!

에이징(Aging) 기법

- 시스템에서 특정 프로세스의 우선순위가 낮아 무한정 기다리게 되는 경우, 한번 양보하거나 기다린 시간에 비례하여 일정 시간이 지나면 우선순위를 한 단계씩 높여 가까운 시간 안에 자원을 할당받도록 하는 기법입니다.
- SJF나 우선순위 기법에서 발생할 수 있는 무한 연기 상태, 기아 상태를 예방할 수 있습니다.

20.6, 19.8, 18.4, 18.3, 17.8, 17.3, 16.8, 15.8, 15.3, 14.5, 14.3, 13.8, 13.6, 13.3, 12.8, 12.5, 11.8, 11.6, 11.3, …

핵심

158 선점 스케줄링의 종류

선점 우선순위	준비상태 큐의 프로세스들 중에서 우선순위가 가장 높은 프로세스에게 먼저 CPU를 할당하는 기법
SRT(Shortest Remaining Time)	비선점 기법인 SJF 알고리즘을 선점 형태로 변경한 기법으로, 현재 실행중인 프로세스의 남은 시간과 준비상태 큐에 새로 도착한 프로세스의 실행 시간을 비교하여 가장 짧은 실행 시간을 요구하는 프로세스에게 CPU를 할당하는 기법
RR(Round Robin)	• 시분할 시스템(Time Sharing System)을 위해 고안된 방식으로, FCFS(FIFO) 알고리즘을 선점 형태로 변형한 기법

	• FCFS 기법과 같이 준비상태 큐에 먼저 들어온 프로세스가 먼저 CPU를 할당받지만 각 프로세스는 할당된 시간(Time Slice, Quantum) 동안만 실행한 후 실행이 완료되지 않으면 다음 프로세스에게 CPU를 넘겨주고 준비상태 큐의 가장 뒤로 배치됨 • 할당되는 시간이 클 경우 FCFS 기법과 같아지고, 할당되는 시간이 작을 경우 문맥 교환 및 오버헤드가 자주 발생됨 • 시간 할당량이 너무 작으면 스래싱에 소요되는 시간의 비중이 커짐 • 대화식 시스템에 유용함
다단계 큐(Multi level Queue)	프로세스들을 우선순위에 따라 시스템 프로세스, 대화형 프로세스, 일괄 처리 프로세스 등으로 상위, 중위, 하위 단계의 단계별 준비 큐를 배치하는 CPU 스케줄링 기법
다단계 피드백 큐(Multi level Feedback Queue)	• 특정 그룹의 준비상태 큐에 들어간 프로세스가 다른 준비상태 큐로 이동할 수 없는 다단계 큐 기법을 준비상태 큐 사이를 이동할 수 있도록 개선한 기법 • 특정 큐에서 오래 기다린 프로세스나 I/O 작업 주기가 큰 프로세스 또는 Foreground 큐에 있는 프로세스는 우선순위가 높은 단계의 준비 큐로 이동시키고, CPU의 점유 시간이 긴 작업은 우선 순위가 낮은 하위 단계의 준비 큐로 이동시킴 • 마지막 단계 큐에서는 작업이 완료될 때까지 RR 스케줄링 기법을 사용함

19.8, 19.3, 18.8, 18.4, 17.8, 17.5, 15.8, 15.5, 15.3, 14.5, 13.3, 12.5, 11.8, 11.6, 11.3, 10.9, 10.3, 09.8, 08.9, …

핵심

159 임계 구역/상호 배제

임계 구역(Critical Section)

- 다중 프로그래밍 운영체제에서 여러 개의 프로세스가 공유하는 데이터 및 자원에 대하여 어느 한 시점에서는 하나의 프로세스만 자원 또는 데이터를 사용하도록 지정된 공유 자원(영역)이다.
- 임계 구역에는 하나의 프로세스만 접근할 수 있으며, 해당 프로세스가 자원을 반납한 후에만 다른 프로세스가 자원이나 데이터를 사용할 수 있다.
- 임계 구역 내로 프로세스가 진입하는 것을 허용하는 것은 운영체제의 제어 권한이다.
- 특정 프로세스가 독점해서는 안되며, 임계 구역 내에서의 작업은 신속하게 진행되어야 한다.
- 임계 구역에서는 프로세스가 무한 루프에 빠지지 않도록 해야 한다.
- 두 개 이상의 프로세스들이 임계 구역을 동시에 사용함으로써 프로세스들이 아무것도 하지 못하게 되는 임계 영역 문제가 발생할 수 있는데, 이를 해결하기 위해서는 상호 배제(Mutual Exclusion), 진행(Process), 한계 대기(Bounded Waiting)라는 3가지 조건을 충족해야 한다.

상호 배제(Mutual Exclusion)

- 특정 프로세스가 공유 자원을 사용하고 있을 경우 다른 프로세스가 해당 공유 자원을 사용하지 못하게 제어하는 기법이다.

- 여러 프로세스가 동시에 공유 자원을 사용하려 할 때 각 프로세스가 번갈아가며 공유 자원을 사용하도록 하는 것으로 임계 구역을 유지하는 기법이다.
- 상호 배제 기법을 구현하기 위한 방법에는 소프트웨어적인 방법과 하드웨어적인 방법이 있다.

소프트웨어적 구현 방법	데커(Deker) 알고리즘, 피터슨(Peterson) 알고리즘, 빵집 알고리즘
하드웨어적 구현 방법	Test & Set 기법, Swap 명령어 기법

19.8, 19.4, 18.3, 17.5, 17.3, 16.8, 09.5, 06.3, 03.8, 03.3, 01.6, 01.3, 00.7, 99.6

핵심

160 세마포어(Semaphore)

- 동기화를 구현할 수 있는 기법 중 하나로, 각 프로세스에 제어신호를 전달하여 순서대로 작업을 수행하도록 하는 기법이다.

잠깐만요! 동기화(Synchronization)

두 개 이상의 프로세스를 한 시점에서는 동시에 처리할 수 없으므로 각 프로세스에 대한 처리 순서를 결정하는 것으로, 상호 배제의 한 형태입니다.

- 다익스트라(E. J. Dijkstra)가 제안하였으며, 사용되는 연산에는 P와 V, 초기치 연산(Semaphore Initialize)이 있고, 상호 배제의 원리를 보장한다.
- S는 P와 V 연산으로만 접근 가능한 세마포어 변수로, 공유 자원의 개수를 나타낸다.
- **세마포어의 종류**
  - **이진 세마포어(Binary Semaphore)** : 세마포어 변수가 0 또는 1을 가지며, 1 이상의 정수로 초기화 됨
  - **산술 세마포어(Counting Semaphore)** : 세마포어 변수가 0 또는 양의 정수를 가지며, 0 이상의 정수로 초기화 됨
- **P 연산** : 자원을 사용하려는 프로세스들의 진입 여부를 자원의 개수(S)를 통해 결정하는 것으로, Wait 동작이라고 함
- **V 연산** : 대기중인 프로세스를 깨우는 신호(Wake Up)로서, Signal 동작이라고 함