1주차 1차시

○ 들어가기

학습목표

- 운영체제의 개념과 정의를 설명할 수 있다.
- 운영체제의 목적으로 처리능력 향상, 반환시간 감소, 사용가능도 향상, 신뢰도 향상 등을 들어 서술할 수 있다.
- 운영체제의 역할에 대하여 여러 가지 예를 들어 말할 수 있다.

주요용어

-처리능력(Throughput) : 작업이 진행되면서 일정한 시간 안에 컴퓨터 시스템이 처리할 수 있는 일의 양
-반환시간(Turn-around time) : 컴퓨터 시스템을 이용하여 작업을 의뢰한 시작 시간부터 처리가 완료되는 시간까지 처리하는데 걸리는 시간
-사용 가능도(Availability) : 컴퓨터 시스템을 이용하려고 시도할 때 컴퓨터 시스템이 얼마나 빨리 사용 가능할 수 있는가를 나타내는 정도
-신뢰도(Reliability) : 컴퓨터 시스템이 작업을 끝냈을 때 얼마나 정확하게 해결을 하였는가를 나타내는 정도

소주제1

1. 운영체제의 개념과 정의

1.1 운영체제

- 컴퓨터 시스템의 여러 가지 각종 자원을 효율적으로 관리 운영하여 사용자로 하여금 컴퓨터 시스템을 사용하는데 편리함을 제공하고 각종 위험에 효과적으로 대처할 수 있는 환경을 제공하는 프로그램들이다. 즉, 사용자와 컴퓨터 사이에 인터페이스를 형성하여 시스템 프로그램을 효과적으로 운용할 수 있도록 하는 시스템 프로그램이다.

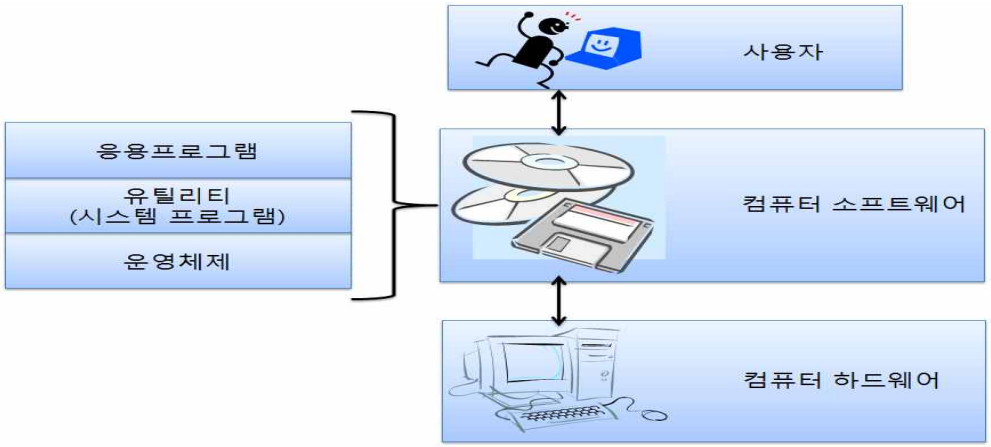
1.2 시스템 소프트웨어

- 컴퓨터 시스템 전체를 운용하는 소프트웨어로써 프로그램의 흐름을 관장하는 역할을 한다.

- 프로그램을 주기억장치에 적재
- 인터럽트
- 언어번역
- 컴퓨터 내부의 여러 장치 관리

운영체제는 시스템 소프트웨어 중의 하나이다. 엄격하게 말하자면 운영체제는 시스템 소프트웨어를 구성하는 일부분으로 설명된다. 컴퓨터 시스템을 운용하는 프로그램이라는 기능으로 볼 때 시스템 소프트웨어 중의 가장 대표적인 프로그램으로 볼 수 있다.

※ 컴퓨터 시스템의 구성



소주제2

2. 운영체제의 목적

2.1 처리능력 향상

- 처리능력 (Throughput)
- 작업이 진행되면서 일정한 시간 안에 컴퓨터 시스템이 처리할 수 있는 일의 양. (수치가 높을수록 좋다.)

2.2 반환시간 감소

- 반환시간 (Turn-around time)
- 컴퓨터 시스템을 이용하여 작업을 의뢰한 시작 시간부터 처리가 완료되는 시간까지 처리하는데 걸리는 시간. (수치가 낮을수록 좋다.)

2.3 사용 가능도 향상

- 사용 가능도 (Availability)
- 컴퓨터 시스템을 이용하려고 시도할 때 컴퓨터 시스템이 얼마나 빨리 사용 가능할 수 있는가를 나타내는 정도. (수치가 낮을수록 좋다.)

2.4 신뢰도 향상

- 신뢰도 (Reliability)
- 컴퓨터 시스템이 작업을 끝냈을 때 얼마나 정확하게 해결을 하였는가를 나타내는 정도. (정확도가 높을수록 좋다.)

소주제3

2. 운영체제의 역할

- 컴퓨터 시스템의 성능을 최대로 발휘할 수 있도록 관리하는 기능이다.

- 부팅
- 저장 매체 관리
- 사용자 인터페이스
- 컴퓨터 자원 관리
- 파일 관리
- 프로세스 관리

2.1 부팅

- 부트스트랩(Bootstrap)의 약자
- 컴퓨터 시스템을 시동하거나 재시동 하는 작업

2.2 저장 매체 관리

- 하드디스크와 플로피디스크의 데이터를 기록하거나 읽고, 쓰기 작업하는 과정의 관리
- 파일 할당 테이블(FAT. File Allocation Table) 이용

2.3 사용자 인터페이스 제공

- 운영체제를 제어하기 위한 사용자와 운영체제 사이의 대화(통신)를 제공하는 역할

- 사용자 인터페이스
- CUI(Character User Interface) : 실행 명령어를 키보드로 입력
- GUI(Graphic User Interface) : 실행 명령어를 윈도우에서 입력
- 메뉴 인터페이스 : 마우스 혹은 커서를 이용하여 메뉴 선택
- 아이콘 : 작은 그림 모양으로 실행 선택

2.4 컴퓨터 자원 관리

- 컴퓨터 시스템은 응용프로그램이 사용하는 서로 다른 종류의 많은 하드웨어로 구성
- 자원관리는 운영체제의 중심을 이루는 슈퍼바이저(Supervisor) 또는 커널(Kernel)이 수행
- 슈퍼바이저(Supervisor)
- CPU제어
- 컴퓨터 시스템이 켜져 있는 동안 주기억장치에 상주 하면서 각종 응용프로그램을 관리

2.5 파일 관리

- 복사
- 백업
- 삭제하기
- 이름수정

2.6 프로세스 관리

- 다양한 작업을 동시에 처리할 때 운용하는 기법

① 시분할 시스템
   시간을 나누어 할당하여 프로그램 실행
② 다중 프로그래밍
   여러 개의 프로그램을 하나의 프로세서에서 동시에 실행 하는 기법
③ 다중처리
   다수의 프로세서를 이용하여 하나 혹은 여러 개의 프로그램을 실행하는 기법
④ 분산처리
   컴퓨터 시스템이 각각의 운영체제와 메모리를 가지고 독립적으로 떨어져 있을 때 네트워크로 연결하여 하나의 컴퓨터 시스템처럼 운용하는 기법

1주차 2차시

○ 들어가기

학습목표

- 운영체제의 구성을 이해하여 설명할 수 있다.
- 운영체제의 구성에 따르는 기능을 학습하여 자원관리에 활용할 수 있다.
- 운영체제에서 제공되는 서비스를 알아보고 시스템에 적용할 수 있다.

주요용어

-감시 프로그램(Supervisor) : 각종 프로그램의 실행과 처리등 프로그램의 흐름 전체를 관리·감독하고 제어
-자원관리 기능 : 컴퓨터 시스템을 구성하는 CPU, 기억장치, 주변장치, 프로그램, 정보, 자료 같은 컴퓨터 자원들을 관리하는 기능
-인터페이스(Interface) : 운영체제와 사용자간의 통신 및 대화를 하기 위한 서비스

소주제1

1. 운영체제의 구성

1.1 제어(Control) 프로그램

(1) 감시 프로그램(Supervisor)
- 가장 중요한 역할
- 각종 프로그램의 실행과 처리등 프로그램의 흐름 전체를 관리·감독하고 제어한다.

(2) 작업 관리(Job Management) 프로그램
- 작업이 수행되도록 준비를 한다.
- 작업을 수행하도록 지시한다
- 작업이 끝나면 마무리한다.
- 다른 작업으로의 이동을 처리한다.
- 작업이 연속적으로 이루어질 수 있도록 시스템 스케줄 및 자원을 할당한다.

(3) 데이터 관리(Data Management) 프로그램
- 입출력 데이터를 관리한다.
- 주기억장치와 보조기억장치 사이의 데이터 전송과 수정, 삭제, 보관 등 데이터의 유지보수 기능을 제공한다.

(4) 통신 관리(Communication Management) 프로그램.
- CPU와 연결된 각 주변장치 간의 신호교환이 원활히 이루어지게 통제한다.
- 외부 통신 회선과 연결된 통신망의 통신제어를 담당한다.

1.2 처리(Process) 프로그램

(1) 언어 번역 프로그램
- 저급언어, 고급언어로 작성한 프로그램을 기계어로 번역하는 기능을 제공한다.
- 컴파일러, 어셈블러, 인터프리터 등

(2) 서비스 프로그램
컴퓨터 시스템을 효율적으로 사용할 수 있도록 지원하는 사용 빈도가 높은 프로그램.
- 연결 편집 프로그램/링커
- 로더
- 정렬/병합 프로그램
- 유틸리티 프로그램

(3) 문제 프로그램

- 특정 업무 및 해결을 위해 사용자가 작성한 프로그램
- 편집기, 데이터베이스, 통신용 프로그램, 그래픽 프로그램 등 사용자의 응용처리를 위해 사용되는 프로그램

소주제2

2. 운영체제의 기능

2.1 자원관리 기능

- 컴퓨터 시스템을 구성하는 CPU, 기억장치, 주변장치, 프로그램, 정보, 자료 같은 컴퓨터 자원들을 관리하는 기능.

① 프로세스 관리 기능
- 프로세스와 스레드 스케줄링
- 프로세스 생성과 제거
- 프로세스의 시작, 정지, 재수행
- 프로세스 동기화 및 통신 관리
- 주기억장치 관리를 위해 주기억장치 관리자와 협력

② 기억장치(주기억장치, 보조기억장치) 관리 기능
- 메모리 상태 추적 및 기억
- 메모리 할당 및 회수
- 가상기억장치 및 페이징 장치 관리
- 장치 관리자 또는 파일 관리자와 협력

③ 장치(입·출력) 관리 기능
- 입·출력 장치의 스케줄링 및 관리
- 각종 주변장치의 스케줄링 및 관리

④ 파일 관리 기능
- 파일의 생성과 삭제, 변경, 유지들의 관리
- 정보의 위치, 사용여부와 상태 등을 추적 관리

소주제3

3. 운영체제의 서비스

- 프로그램을 실행하기 위한 환경을 제공하고, 프로그램과 사용자들에게 정해진 서비스를 제공한다. 운영체제마다 제공되는 서비스가 다르다.

3.1 부트스트랩 서비스

- 부트스트래핑(Bootstraping), 부팅(Booting)
- 운영체제가 적재되는 과정
- 부트스트랩 로더는 디스크 트랙 0에 나머지는 디스크의 다른 부분에 적재

3.2 사용자 서비스

프로그래머가 프로그래밍 작업을 쉽게 수행 할 수 있도록 제공

① 사용자 인터페이스

- 운영체제와 사용자간의 통신 및 대화를 하기 위한 서비스

- GUI(Graphic User Interface)
- CUI(Chracter User Interface)
- CLI(Command Line Interface)

② 프로그램 수행

- 프로그램을 메모리에 적재하여 실행한다.
- 프로그램을 정상적 또는 비정상적으로 끝낼 수 있어야 한다.

③ 입·출력 동작

- 수행 중인 프로그램은 입출력을 요구할 수 있다.
- 이러한 입출력은 파일 또는 입출력 장치를 지정할 수 있다.

④ 파일 시스템 조작

- 프로그램은 파일을 정확히 읽고 기록해야 한다.
- 파일을 생성, 삭제 할 수 있어야 한다.

⑤ 통신

- 하나의 프로세스와 또 다른 프로세스 사이의 정보 교환을 위한 통신 제공
- 같은 컴퓨터 시스템 내부에서 수행되는 프로세스 사이에 일어나는 통신
- 네트워크로 연결된 각각 다른 컴퓨터 시스템 간에 수행되는 프로세스의 정보 교환을 위한 통신

⑥ 오류 발견

- 컴퓨터 시스템의 모든 장치에서 일어나는 오류를 탐지 할 수 있어야 한다.

3.3 시스템 서비스

① 자원 할당

- 다수의 사용자 또는 다수의 작업이 동시에 실행될 때 데이터 혹은 자원들이 각각의 작업에 할당되어야 한다.

② 보호

- 다중 사용자 컴퓨터 시스템에 저장된 정보의 소유자는 다른 사람이 자신의 정보에 접근 하는 것을 제한 할 수 있다.

③ 계정

- 다중의 사용자가 컴퓨터 시스템을 사용하는 기록을 보관한다.
- 컴퓨터 시스템 사용량을 알거나, 시스템 사용료 청구, 시스템 사용 통제 등의 관리를 할 수 있다.

3.4 시스템 호출 서비스

① 실행중인 프로그램과 운영체제 간의 인터페이스이다.
② 시스템 호출을 통해 운영체제의 기능을 서비스 받는다.
③ API(Application Programming Inteface)

- 프로세스 제어
- 파일조작
- 장치조작
- 정보관리
- 통신

1주차 3차시

○ 들어가기

학습목표

• 반도체 소자에 따른 운영체제의 발전과정에 대해서 학습하여 서술할 수 있다.
• 운영체제의 종류에 대해서 알고 종류별로 구분할 수 있다.

주요용어

-집적회로(IC) : 여러 개의 트랜지스터(TR)를 결합하여 회로를 구성한 것.
-고밀도집적회로(LSI) : 여러 개의 집적회로(IC)를 결합하여 집적도를 높인 회로.
-초고밀도집적회로(VLSI) : 고밀도집적회로(LSI)보다 집적도를 높인 회로.

소주제1

1. 운영체제의 발달과정

- 컴퓨터를 구성하는 반도체 소자의 발전에 따라 운영체제도 함께 발전하였다. 현대의 반도체 소자는 구조적으로 많은 발전을 하였다.

- 제1세대(진공관(Tube), 1940년대~1950년대)
- 제2세대(트랜지스터(Transistor), 1950년대~1960년대)
- 제3세대(트랜지스터(Transistor), 집적회로(IC), 1960년대~1980년대)
- 제4세대(고밀도집적회로(LSI), 초고밀도집적회로(VLSI), 1980년대~1990년대)
- 제5세대(초고밀도집적회로(VLSI), 1990년대~현재)

1.1 제1세대

- 1940년대 중반 ~ 1950년대 중반
- 진공관(Tube) 사용
- 기계어 사용
- 기계어로 프로그램 작성
- 운영체제의 의미가 없음

1.2 제2세대

- 1950년대 중반 ~ 1960년대 중반
- 트랜지스터(Transistor) 사용
- 일괄처리 시스템 등장
  - 버퍼링
  - 스풀링
- 작업제어 언어 등장
- 한 작업에서 다른 작업으로의 전환 자동처리
- 입출력 제어 시스템 등장
- 기계어, 어셈블리어 사용

1.3 제3세대

- 1960년대 중반 ~ 1980년대 초반
- 트랜지스터(Transistor), 집적회로(IC) 사용
- 운영체제 고급언어로 작성
  - UNIX
- 장치 독립성 제공
- 다중모드 : 일괄처리, 시분할처리, 실시간 처리, 다중처리 제공
- 작업제어 언어 복잡
- 소프트웨어 계층구조 개념, 소프트웨어공학 등장
- 다중 프로그래밍 : 다중 사용자를 위한 CPU작업 할당
- 다중처리 : 다중 프로세스를 위한 시스템 처리능력 향상

- 시분할 시스템 : 사용자와 컴퓨터 인터페이스 운영으로 시분할 시스템 등장

1.4 제4세대
- 1980년대 초반 ~ 1990년대 초반
- 고밀도집적회로(LSI), 초고밀도집적회로(VLSI) 사용
- 컴퓨터 네트워크와 온라인 처리 사용
- 마이크로프로세서 등장 : PC 개발
- 가상기계, 데이터베이스 시스템 등장
- 분산 데이터 처리 개념
- UNIX, DOS, VMS 사용
- Windows 개발
- Workstation, Multi-Processor 일반화

1.5 제5세대
- 1990년대 초반 ~ 현재
- 초고밀도집적회로(VLSI), 조셉슨 소자, 칼륨-비소 소자, 광-소자 사용
- 네트워크 시스템, 분산처리 시스템 실용화
- 지식기반 시스템 등장
- 인공지능 실현
- 논리, 추론 기능 강화
- 가상 머신 개념 등장
- 사용자와 컴퓨터간 대화 기능 실현
- 정보기술 융합 실현
- GUI 등 사용자 위주의 시스템

소주제2

2. 운영체제의 종류

- 많은 종류의 운영체제가 있지만 가장 많이 사용하는 대표적인 운영체제는 다음과 같다.

- UNIX
- Linux
- Windows
- Macintosh
- DOS

2.1 서버용 운영체제

- DOS, DR-DOS, MS-DOS, FreeDOS, K-DOS, IBM-DOS
- UNIX
    - BSD
        - FreeBSD
        - XNU, iOS, macOS, iPadOS
    - System V, 솔라리스(SmartOS), HP-UX
    - Redox, VMware ESXI(vSphere, vCenter)
    - Linux
        - 데비안, 우분투, 라즈베리 파이 OS
        - 페도라, Red Hat, CentOS, MeeGo
        - 슬랙웨어
        - 아치
        - 안드로이드, Remix Linux, Polaris OS
        - 젠투, 크롬 OS
        - Solus
- Windows
    - Windows 9x
    - Windows NT, Windows 20xx
    - OS X Server
- Mac OS, OS/2

2.2 개인용 운영체제

- IBM-DOS, MS-DOS, Windows95, Windows98, Windows ME, Windows XP, Windows Vista, Windows 7, Windows 8, Windows 10, Windows 11, Mac OS.

2.3 모바일 운영체제

- iOS, 안드로이드, 심비안 OS, 블랙베리 OS, 윈도우 폰8.1, 리눅스 OS, 웹 OS, BREW, 바다 OS, Tizen, Maemo, Moblin, Meego, LiMo,

2.4 단일 작업 처리는 DOS를 사용

- CUI(Character User Interface)
- CLI(Command Line Interface)

2.5 다중 작업 처리는 Windows를 비롯하여 UNIX, Linux 사용

- GUI(Graphic User Interface)
- CUI(Character User Interface)
- CLI(Command Line Interface)

2주차 1차시

○ 들어가기

학습목표

- 운영체제의 프로세스 수행방식에 따라 여러 가지로 나누어서 구분할 수 있다.
- 프로세스의 진행에 따라 분류되는 시스템의 종류와 특징에 대해 학습하여 설명할 수 있다.

주요용어

- 일괄처리 시스템 - 비슷한 여러 개의 작업을 모아서 한꺼번에 처리하는 방식
- 다중프로그래밍 - 하나의 CPU를 이용하여 여러 개의 프로그램을 동시에 처리하는 방식
- 다중처리 시스템 - 여러 개의 CPU를 이용하여 여러 개의 프로그램을 동시에 처리하는 방식
- 분산처리 시스템 - 지역적으로 분산된 여러 대의 컴퓨터를 네트워크 연결하여 작업을 처리하는 방식

소주제1

1. 오프라인 시스템

1.1 일괄처리 시스템 (Batch Processing System)

① 대부분의 오프라인 시스템에서 활용
② 초기 컴퓨터 시스템에서 사용된 방식
③ 작업준비 시간을 절약하기 위하여 비슷한 여러 개의 작업을 모아서 한꺼번에 처리하는 방식
④ 상주모니터를 활용
⑤ 일괄처리를 위한 작업제어카드(JCC) 필요
⑥ 자기테이프를 주로 사용
⑦ 프로그램 오류 수정 작업의 어려움
⑧ 컴퓨터 시스템을 효율적으로 사용
⑨ 반환시간이 늦음
- 작업처리 할 때는 CPU 유휴시간 감소
- 작업처리가 없을 때는 CPU 유휴시간이 길어짐

⑩ 급여계산, 은행 결산 등의 업무에 사용

[그림] 오프라인-일괄처리 시스템

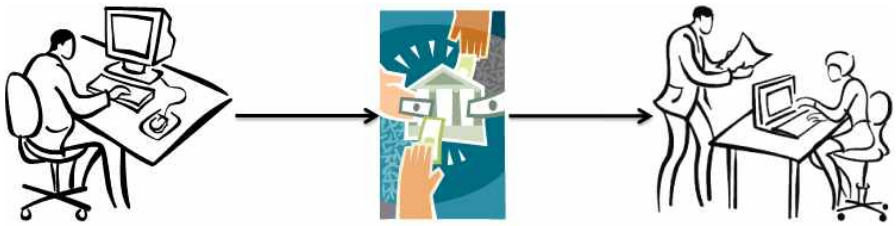
소주제2

2. 온라인 시스템

2.1 일괄처리 시스템 (Batch Processing System)

① 온라인으로 연결하여 비슷한 여러 개의 작업을 모아서 한꺼번에 처리하는 방식
② 일괄처리를 위한 작업제어언어가(JCL) 필요
③ 프로그램 오류 수정 작업의 어려움
④ 컴퓨터 시스템을 효율적으로 사용
⑤ 반환시간이 늦음
- 작업처리 할 때는 CPU 유휴시간 감소
- 작업처리가 없을 때는 CPU 유휴시간이 길어짐

⑥ 급여계산, 은행 결산 등의 업무에 사용



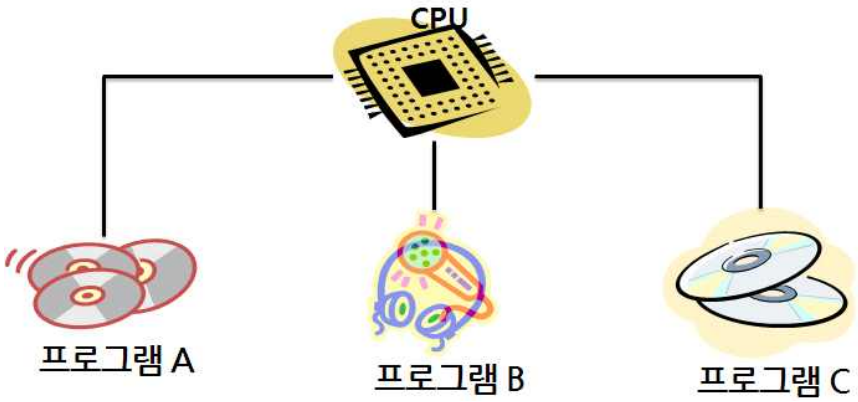
[그림] 온라인-일괄처리 시스템

2.2 다중프로그래밍 시스템(Multi-Programming System)

- 하나의 CPU를 이용하여 여러 개의 프로그램을 동시에 처리하는 방식이다.

① 하나의 주기억장치에 여러 개의 프로그램을 저장하고 동시에 프로그램을 수행
② CPU의 유휴시간 감소시킴
- CPU 사용률 증가
- 프로그램 처리속도 향상
- 프로그램 처리량 증가

③ 기억장치 관리 기법 필요
④ CPU 스케줄링 기법 필요



[그림] 다중프로그래밍 시스템

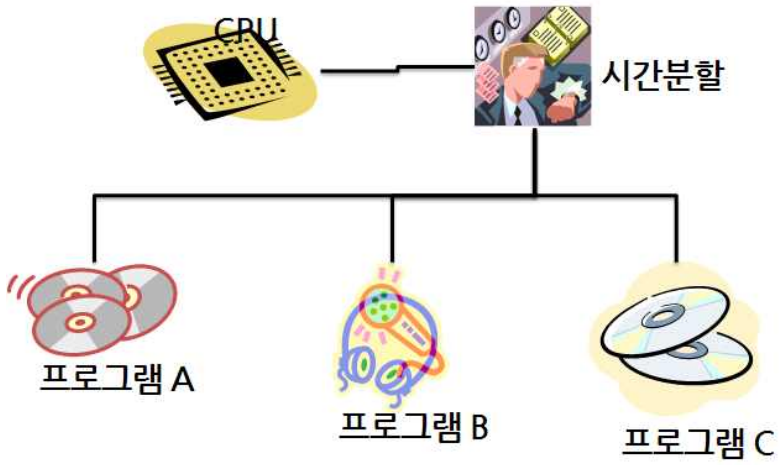
2.3 시분할 시스템(Time Sharing System)

- 한 대의 컴퓨터를 여러 명의 사용자가 사용하면서, 각자 독립된 컴퓨터를 사용하는 것처럼 사용하도록 하는 것이다.

① CPU를 여러 명이 공동으로 사용

② 하나의 CPU가 여러 프로그램을 동시에 수행불가

- CPU 사용시간을 일정시간으로 강제할당
- CPU 스케줄링 기법에 따라 CPU 사용 순서 결정



[그림] 시분할 시스템

2.4 다중처리 시스템(Multi-Processing System)

- 여러 개의 CPU를 이용하여 여러 개의 프로그램을 동시에 처리하는 방식이다.

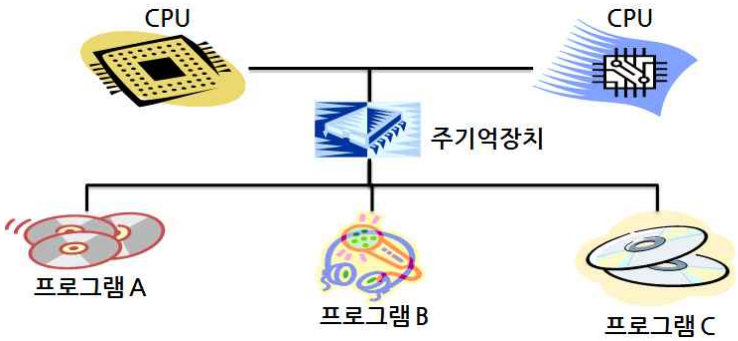
① 하나의 CPU가 고장 나더라도 업무수행 가능

- 시스템의 안정성 증가
- 시스템의 신뢰도 증가

② 여러 개의 CPU 공유 및 스케줄링 결정 요구

- 최적의 연결기법
- 기억장치, I/O장치 등의 자원 공유 문제

③ 강 결합 다중처리 시스템
- CPU간 매우 밀접하게 동기화
④ 약 결합 다중처리 시스템
- CPU간 밀접 보다 작업 처리량 향상에 중점



[그림] 다중처리 시스템

2.5 실시간 처리 시스템(Real Time Processing System)

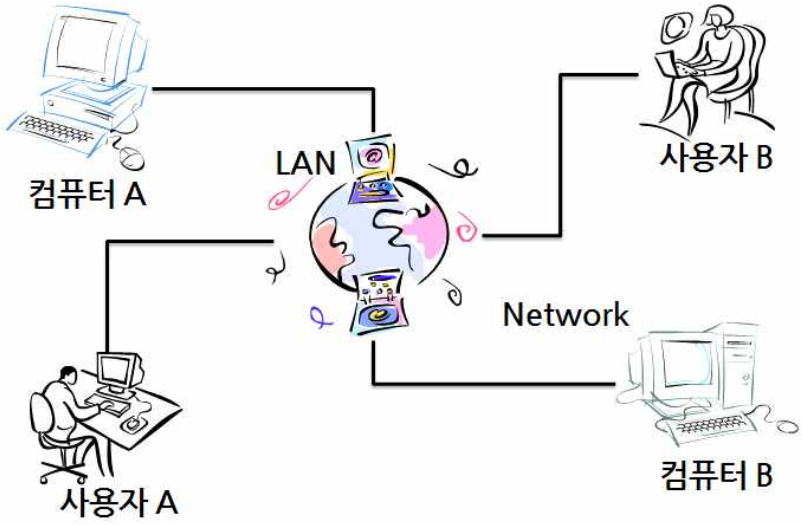
- 데이터 발생 후 즉시 처리하는 방식이다.

① 처리시간 단축
② 처리비용 절감
③ I/O 데이터 임시저장 장소 필요
④ 시스템 장애 시 업무 마비
⑤ 레이더 추적, 은행 업무, 호텔 예약, 교통시스템 좌석 예약, 가상현실, 로봇제어 등에 사용

2.6 분산 처리 시스템(Distributed Processing System)

- 지역적으로 분산된 여러 대의 컴퓨터를 네트워크 연결하여 작업을 처리하는 방식이다.

① 각각의 컴퓨터 운영체제는 같지 않아도 됨
② 자원 공유
③ 처리속도 증가
④ 신뢰도 향상
⑤ 네트워크 통신 기능



[그림] 분산처리시스템

2.7 다중모드 처리 시스템(Multi-Mode Processing System)

- 일괄처리 시스템, 시분할 시스템, 다중처리 시스템, 실시간 처리 시스템을 모두 제공하는 시스템이다.

2.8 병렬처리 시스템(Parallel Processing System)

- 둘 이상의 프로세서를 이용하여 작업을 동시에 처리하는 시스템이다.
- 여러 개의 프로세서를 통해서 각각 작업을 수행함으로써 단위시간당 처리량을 높일 수 있어서 처리능력이 향상된다.

2주차 2차시

○ 들어가기

학습목표

• 저급언어를 서술할 수 있다.
• 고급언어를 설명할 수 있다.
• 언어 번역 프로그램의 종류 및 특징을 비교분석할 수 있다.

주요용어

-기계어 : 컴퓨터 시스템을 위하여 직접 사용되는 언어.
-어셈블리어 : 기계어 또는 컴퓨터 시스템을 제어하기 위한 언어.
-고급언어 : '자연어'라고 일컫는 인간이 사용하는 언어와 비슷한 형태의 언어.

소주제1

1. 저급언어

1.1 기계어

- 컴퓨터 시스템을 위하여 직접 사용되는 언어이다.

- 컴퓨터가 직접 이해 가능한 언어 사용
- 사용자가 이해하고 작성하기 어려움
- 2진수 형태로 표현되고 프로세스 시간이 빠름
- CPU 내장 명령어를 직접 사용
- 컴퓨터 시스템 마다 각각의 기계어를 사용하는 경우가 많으므로 컴퓨터 시스템 간의 언어 호환이 어려움

1.2 어셈블리어

- 기계어 또는 컴퓨터 시스템을 제어하기 위한 언어이다.
- 명령어를 쉽게 연상할 수 있는 기호를 기계어와 1:1로 연관시켜 만든 언어
- 니모닉(Mnemonic, 연상언어/상징언어)언어
- 컴퓨터 시스템이 직접 이해하기 어려운 기계어에 가까운 언어
- 사용자가 직접 이해 가능한 언어
- 컴퓨터 시스템 마다 다른 기계어를 사용하므로 호환하기 어려움
- 기계어로 번역을 위한 “어셈블러”라는 언어번역 프로그램으로 번역해야 함
- 어셈블리어 명령

⇒ [의사명령어] 원시프로그램을 어셈블 할 때 어셈블러가 해야 할 동작을 지시

- START, END, USING, DROP, EQU…등

⇒ [실행명령] 데이터 처리 명령

- A, AH, AR, S, SR, L, LA, ST, C, BNE

- 어셈블리어 명령어 형식

→ Label, OP, Operand 로 구성

Label : 데이터를 기억할 장소, 분기할 위치, 기호상수 등에 대한 기호(Symbol)를 기술하는 부분 (생략 가능)
OP : 명령어(OP-code)를 기술하는 부분
Operand : 명령어(OP-code)가 연산을 수행하기 위한 연산의 대상이 되는 Literal (상수, 데이터)이나 주소, Register 번호 등을 기술하는 부분

※ 어셈블리어 명령어 형식 (예)

- DATA1 기억장소에 있는 데이터 2000개의 합을 구하여 SAVE 기억장소에 저장하는 프로그램

〈Label〉	〈OP〉	〈Operand〉
PR2	START	
	USING	*,15

	LA	15,*
	SR	TOTAL, TOTAL
	SR	3, 3
AA	L	2, DATA1(3)
	AR	TOTAL, 2
	A	3, F‘1’
	C	3, F’2000’
	BNE	AA
	ST	TOTAL, SAVE
TOTAL	EQU	4
SAVE	DS	
DATA1	DC	F’25, 26, 97, 101 …’
		(2000 NUMBER)
	END	

소주제2

2. 고급언어

- ‘자연어’라고 일컫는 인간이 사용하는 언어와 비슷한 형태의 언어이다.

- 자연어를 기계어로 번역하기 위한 “컴파일러” 혹은 “인터프리터”라는 언어번역 프로그램이 필요한 언어
- “컴파일러 언어”라고도 함
- 프로그램의 흐름을 충분히 파악한다면 프로그램의 작성과 수정이 쉬움
- 저급언어를 제외한 사용자가 사용하는 프로그래밍언어 대부분이 고급언어

2.1 컴파일러를 사용하는 고급언어

- C, C++, FORTRAN, COBOL, ALGOL, PASCAL, PL/1…등

2.2 인터프리터를 사용하는 고급언어

- BASIC, SNOBOL, LISP, APL…등

소주제3

3. 언어 번역 프로그램

3.1 어셈블러

기계 명령과 절대 주소에 기호를 부여한 어셈블리 언어를 입력받아서 목적 프로그램을 생성하는 번역기
1-패스, 2-패스가 있는데 일반적으로 2-패스로 구성된다.
2-패스 : 첫 패스에서 기호표 만들기, 두 번째 패스에서 기호표를 이용하여 기계어 번역을 완성.

3.2 컴파일러

고급언어로 작성된 프로그램 모듈 전체를 번역하여 목적프로그램을 생성하는 번역기

3.3 인터프리터

고급언어로 작성된 원시 프로그램을 한 문장씩 번역하여 즉시 실행하는 번역기

3.4 매크로 프로세서

매크로 명령문을 해당 어셈블리 언어의 명령문으로 확장시켜주는 프로세서.
매크로 정의 인식, 매크로 정의 저장, 매크로 호출 인식, 매크로 호출 확장, 아규먼트 치환이 있다.

3.5 프리 프로세서

고급언어인 원시 프로그램을 또 다른 고급언어로 번역하는 번역기.

3.6 크로스 컴파일러

한 컴퓨터에서 실행되는 프로그램을 다른 컴퓨터에서 실행될 수 있도록 번역할 때 사용하는 컴파일러.

2주차 3차시

○ 들어가기

학습목표

• 어셈블러에 대해 설명할 수 있다.
• 매크로 프로세서를 정의할 수 있다.
• 컴파일러와 인터프리터를 비교분석할 수 있다.
• 링커와 로더를 학습하여 서술할 수 있다.

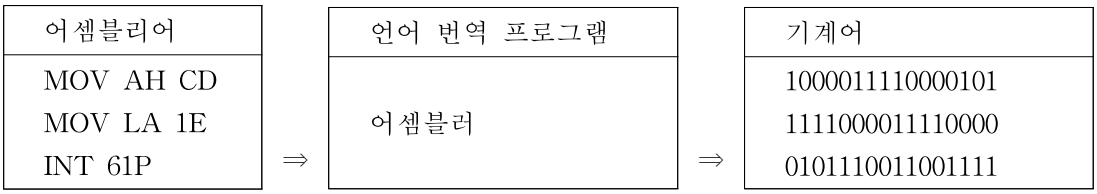
주요용어

-어셈블러 : 어셈블리어로 작성된 원시 프로그램을 기계어로 번역하는 언어 번역 프로그램.
-컴파일러 : 고급언어로 작성된 프로그램을 목적 프로그램으로 번역한 후, 기계어로 된 실행 프로그램으로 번역해주는 언어 번역 프로그램.
-인터프리터 : 고급언어로 작성된 프로그램을 한 줄 단위로 받아들여 번역하고, 동시에 한 줄 단위로 실행시키는 언어 번역 프로그램.
-링커 : 언어 번역 프로그램에 의해 생성된 목적프로그램, 라이브러리, 실행 프로그램 등을 연결해 주는 시스템 소프트웨어

소주제1

1. 어셈블러

- 어셈블리어로 작성된 원시 프로그램을 기계어로 번역하는 언어 번역 프로그램이다.
  ⇒ 어셈블리어로 작성된 원시프로그램을 목적 프로그램으로 번역하는 과정을 “어셈블” 과정이라고 한다.



1.1 어셈블 과정

(1) 단일 패스 어셈블러(Pass-1, One Pass Assembler)
원시 프로그램을 명령문 하나씩 읽어서 기계어로 번역하여 목적 프로그램을 생성한다.

(2) 이중 패스 어셈블러(Pass-2, Two Pass Assembler)
원시 프로그램을 2번 읽는데, 1단계 작업을 수행하기 위하여 한번 읽고 난 후, 2번째 읽으면서 1단계 결과를 이용하여 완전한 목적 프로그램을 생성한다.

1.2 Table의 종류 및 구성

- 기계 명령어 테이블(MOT, Machine Operation Table)
- 의사 명령어 테이블(POT, Pseudo Operation Table)
- 기호 테이블(ST, Symbol Table)
- 리터럴 테이블(LT, Literal Table)

1.3 Pass-1과 Pass-2 과정 비교

구분	Pass-1	Pass-2
목적	기호와 리터럴 정의	기호 번지에 대한 상대 번지를 생성하고, 목적 프로그램 생성
기 능	기계 명령어의 길이 정의 위치 계수기(PC, LC) 관리 기호(Symbol)들의 값을 ST에 기억 사용된 리터럴들을LT에 기억 해당하는 의사 명령어 처리	기계 명령어 생성 ST에서 기호(Symbol)들의값을 찾음 의사 명령어 처리 리터럴 발생
사용관련 데이터베이스	원시 프로그램(Source Program) 위치계수기(PC) MOT, POT, ST, LT	원시프로그램(Source Program)의사본 위치계수기(PC) Pass-1에서만든 ST, LT MOT, POT, 베이스레지스터 테이블 PRINTLINE(어셈블결과 보고서 인쇄) 목적 프로그램(ObjectProgram)

소주제2

2. 매크로 프로세서(Macro Processor)

어셈블리어 + 매크로 ⇒ 매크로 프로세서 ⇒ 어셈블러 ⇒ 기계어

2.1 매크로 프로세서의 역할

- 어셈블리어를 사용하기 쉽게 명령어들을 문자로 치환한다.
- 매크로 라이브러리 : 자주 사용되는 매크로들을 모아놓은 곳이다.
- 문자열 치환처럼 사용된 횟 수 만큼 명령어를 생성•삽입해서 실행한다.
- 매크로 정의 내에 또 다른 매크로 정의를 할 수 있다.
- 파스칼(Pascal) 언어는 매크로 프로세서 기능이 없다.

어셈블리어
MOV AH CD MOV LA 1E INT 61P

=

매크로
SEND “PI”

2.2 매크로 프로세서의 기능

- 매크로 정의 인식
- 매크로 정의 저장
- 매크로 호출 인식
- 매크로 확장 및 인수

소주제3

3. 컴파일러와 인터프리터

3.1 컴파일러

- 언어 번역 프로그램이 고급언어 종류마다 다름
- 고급언어를 이용하여 작성된 프로그램을 번역하고 실행하는 과정
- 고급언어로 작성된 프로그램을 목적 프로그램으로 번역한 후, 링킹과 로더의 작업을 하여 컴퓨터에서 실행 가능한 실행 프로그램(기계어)으로 바꾸어 줌
- 번역하는 과정이 번거롭거나 복잡하고 시간이 오래 걸릴 수 있음
- 컴파일러에 의해 번역이 끝난 실행 프로그램은 실행 속도가 빠름
- 컴파일러를 이용하는 고급언어는 C, C++, FORTRAN, COBOL, ALGOL, PASCAL, PL/1..등이 있음

3.2 인터프리터

- 언어 번역 프로그램이 고급언어 종류마다 다름
- 고급언어를 이용하여 작성된 프로그램을 한 줄 단위로 받아들여서 번역함과 동시에 프로그램을 한 줄 단위로 실행
- 목적 프로그램이 생성되지 않고 프로그램이 직접 실행
- 목적 프로그램을 생성하지 않으므로 번역속도가 빠름
- 원시 프로그램의 수정과 변화에 빠르게 반응함
- 목적 프로그램이 생성되지 않으므로 프로그램을 실행할 때 마다 번역해야 하는 번거로움이 있음
- 프로그램 실행할 때 마다 매번 번역해야 하므로 실행속도가 느림
- 줄 단위로 번역과 실행이 되므로 시분할 시스템에 유용
- 시분할 시스템을 사용하지 않을 경우 CPU 사용시간에 따른 시간 낭비가 큼
- 인터프리터를 이용하는 고급언어는 BASIC, SNOBOL, LISP, APL..등이 있음

3.3 컴파일러와 인터프리터의 비교

구분	컴파일러	인터프리터
번역 단위	전체	행(줄)
목적 프로그램	생성	생성 안함
실행 속도	빠름	느림
번역 속도	느림	빠름
관련 언어	C, C++, FORTRAN, COBOL, ALGOL, PASCAL, PL/1…등	BASIC, LISP, APL, SNOBOL…등

소주제4

4. 링커와 로더

4.1 링커

언어 번역 프로그램에 의해 생성된 목적프로그램, 라이브러리, 실행 프로그램 등을 연결해 주는 시스템 소프트웨어이다.

4.2 로더

실행 프로그램 또는 실행 프로그램에 필요한 정보와 자료를 보조기억장치로부터 주기억장치로 적재하는 시스템 소프트웨어이다.

(1) 로더의 기능

- 할당(Allocation)
- 연결(Linking)
- 재배치(Relocation)
- 적재(Loading

(2) 로더의 종류

- Compile And Go 로더
- 절대 로더(Absolute Loader)
- 직접 연결 로더(Direct Linking Loader)
- 동적 적재 로더(Dynamic Loading loader)

3주차 1차시

○ 들어가기

학습목표

- 프로세스가 무엇인지를 알아보고 정의할 수 있다.
- PCB에 대해 학습하여 설명할 수 있다.
- 프로세스의 종류를 공부하여 종류별 차이점을 분석할 수 있다.

주요용어

- 프로세스 - CPU에 의해서 처리되는 사용자 프로그램과 시스템 프로그램을 의미.
- PCB(Process Control Block) - 프로세스에 대한 중요한 정보를 저장해 놓는 곳.

소주제1

1. 프로세스의 개념과 정의

1.1 프로세스의 개념

- CPU에 의해서 처리되는 사용자 프로그램과 시스템 프로그램을 의미한다.
- 운영체제에 의해서 관리되는 최소 작업 단위를 말한다.

1.2 프로세스의 정의

- PCB(Process Control Block)를 가진 프로그램
- 실행중인 프로그램을 통하여 결과를 얻기 위한 과정
- 프로세서에 의해 프로그램이 진행되는 과정
- 비동기적 행위 (프로세스가 서로 규칙적이지 않고 독립실행)
- 어떤 목적이나 결과에 따라 발생되는 사건들의 과정
- 프로세서가 할당되는 프로그램 혹은 작업

소주제2

2. PCB

- 프로세스 제어 블록(Process Control Block, PCB) 이다.
- 프로세스에 대한 중요한 정보를 저장해 놓는 곳이다.
- Task Control Block, Job Control Block 이라고도 한다.
- 프로세스를 관리하기 위하여 유지
- 프로세스가 생성될 때 PCB가 만들어지고, 프로세스 완료 후 삭제됨
- 메인 메모리에 있으면서 프로세스의 존재 정의

2.1 PCB저장 정보

① 프로세스의 상태
- 생성, 보류, 준비, 실행, 대기, 중단, 교착, 완료 등 표시

② 프로세스의 포인터
- 부모/자식 프로세스 포인터
- 할당된 자원에 대한 포인터

③ 프로세스 스케줄링 정보
- 프로세스의 우선순위 정보

④ 레지스터 저장 정보
- CPU에 저장되는 정보
- 누산기(Accumulator)
- 인덱스/범용 레지스터
- PC(Program Counter)

⑤ 메모리 관리 정보
- 주기억장치
- 페이지 테이블

⑥ I/O 장치 상태 정보
- 입출력 장치
- Open File(개방된 파일)

⑦ 계정 정보
- CPU사용시간, 실제사용시간, 한정시간
- 계정 번호, 프로세스 번호, 작업 번호

소주제3

3. 프로세스 종류

3.1 순차 프로세스

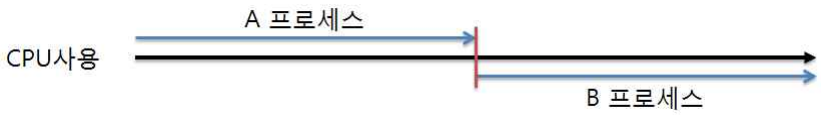
- 현재 실행 중인 하나의 프로세스
- 프로세스 실행은 하나씩 순차적으로 실행함을 원칙으로 한다.
- 어느 시점이든지 하나의 프로세스를 위하여 하나의 명령어가 실행되도록 한다.

3.2 병행 프로세스

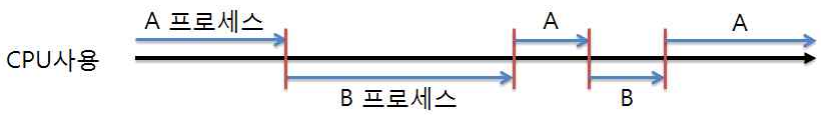
- 현재 실행중인 프로세스가 2개 이상
- CPU가 하나일 때는 CPU 사용을 교대로 한다.
- CPU가 다수일 때는 CPU 사용을 각각의 프로세스를 함께 실행한다.

3.3 한 개의 CPU사용

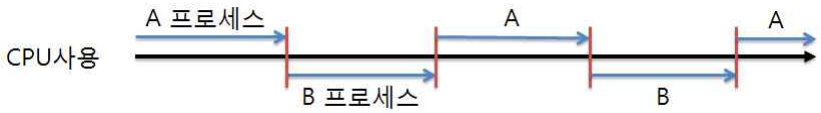
- A프로세스가 끝나고 난 후에 B프로세스를 수행 처리.



- A프로세스와 B프로세스를 CPU 운휴시간에 맞춰서 교대로 수행 처리.

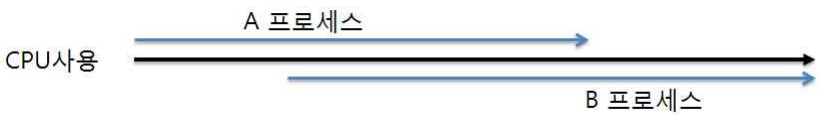


- 시분할 사용으로 인한 프로세스 수행 처리.



3.4 두 개 이상의 CPU 사용

- A프로세스와 B프로세스를 동시에 수행 처리 할 수 있다.



3주차 2차시

○ 들어가기

학습목표

- 프로세스의 진행방법을 설명할 수 있다.
- 프로세스의 상태를 서술할 수 있다.
- 프로세스의 상태전이를 정의할 수 있다.

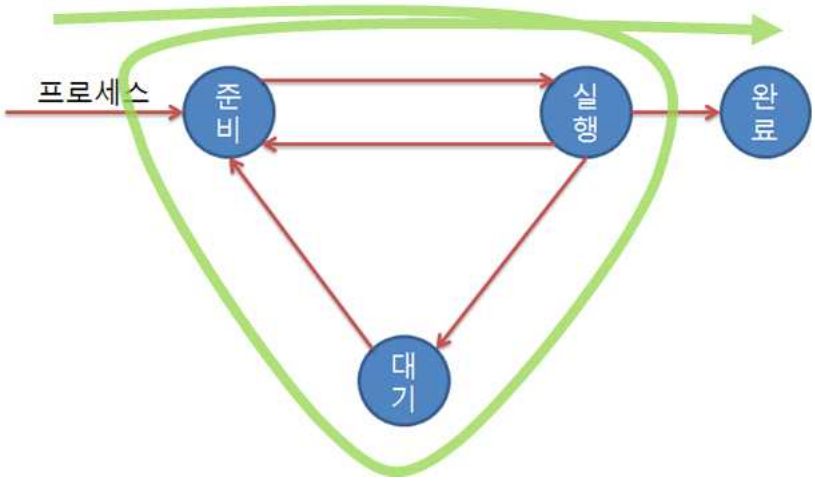
주요용어

- Dispatch - 준비상태 → 실행상태.
- Timeout - 실행상태 → 준비상태
- Block - 실행상태 → 대기상태
- Wakeup - 대기상태 → 준비상태

소주제1

1. 프로세스의 진행

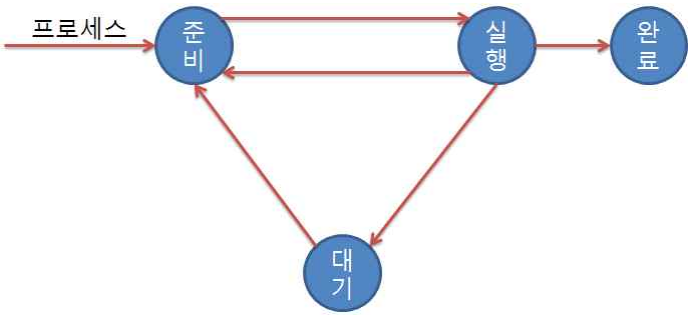
- 준비(Ready)
- 실행(Run)
- 대기(Wait)(보류, 블록)
- 완료(Complete, Terminate)



소주제2

2. 프로세스 상태

- 준비(Ready)상태
- 실행(Run)상태
- 대기(Wait)상태
  - 보류(Pending)상태
  - 블록(Block)상태.
- 교착(Deadlock)상태
- 완료(Complete, Terminate)상태



① 준비(Ready)상태
- 프로세스가 CPU를 할당 받기 위하여 기다리는 상태
- 프로세스가 작업 스케줄링에 의해서 우선순위를 할당된다.

② 실행(Run)상태
- 프로세스가 CPU를 할당 받아서 작업이 진행 중인 상태
- 프로세스 수행 중에 다른 작업이 선점되거나, 시간 분할 또는 스케줄링에 의한 할당시간이 지나면 준비 상태로 전이된다.
- 프로세스 수행 중에 입출력 처리가 필요하면 수행 중인 프로세스는 대기 상태로 전이된다.

③ 대기(Wait)상태
- 프로세스가 실행 중에 입출력 처리가 필요하면 CPU를 비워주고 대기하고 있는 상태
- 대기 상태에서는 작업 스케줄링에 의한 우선순위가 할당되지 않는다.
- 보류(Pending)상태 혹은 블록(Block)상태 라고도 한다.

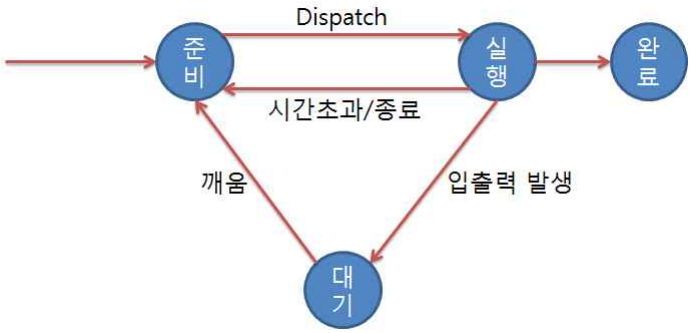
④ 교착(Deadlock)상태
- 프로세스가 서로 엉켜서 수행이 불가능한 상태로 정지되는 상태

⑤ 완료(Complete, Terminate)상태
- 프로세스가 CPU를 할당 받아 주어진 시간 동안 수행을 끝낸 상태

소주제3

3. 프로세스 상태 전이

- Dispatch
  - 준비상태 → 실행상태
- Timeout
  - 실행상태 → 준비상태
- Block
  - 실행상태 → 대기상태
- Wakeup
  - 대기상태 → 준비상태



① Dispatch [준비상태 → 실행상태]

- 준비상태에 있는 프로세스 중에서 작업 스케줄링 우선순위에 따라 하나의 프로세스를 CPU에 할당하고 실행 상태로 전이하는 과정
- CPU 할당시간이 있을 경우 할당시간을 함께 정해준다.

② Timeout [실행상태 → 준비상태]

- 프로세스가 실행 중에 CPU 할당시간이 모두 끝나면 다른 프로세스의 실행을 위하여 CPU를 비워주고 준비상태로 돌아간다.
- CPU 할당시간이 정해지지 않았을 경우에는 작업 스케줄링에 의하여 준비 상태로 돌아간다.

③ Block [실행상태 → 대기상태]

- 프로세스가 실행 중에 입출력 명령을 만나면 인터럽트가 발생되고, 입출력 명령에 의한 프로세스에게 CPU를 비워주고 대기상태로 전이된다.

④ Wakeup [대기상태 → 준비상태]

- 입출력 명령에 의한 프로세스가 완료되면, 대기상태에 있던 프로세스는 실행을 위한 준비상태로 전이된다.

3주차 3차시

○ 들어가기

학습목표

- 스레드에 대한 개념을 알고 설명할 수 있다.
- 스레드의 종류를 구분할 수 있다.
- 스레드의 상태를 학습하여 서술할 수 있다.
- 스레드의 장점을 분석할 수 있다.

주요용어

- 스레드(Thread) - 프로세스 내에서 프로그램을 실행하는 작업 단위.
- 단일 스레드(Thread) - 하나의 프로세스에 한 개의 스레드가 존재.
- 다중 스레드(Thread) - 하나의 프로세스에 여러 개의 스레드가 존재.

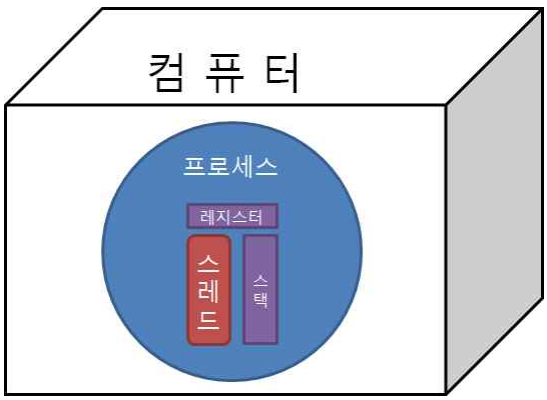
소주제1

1. 스레드의 개요

1.1 스레드(Thread)란?

- 프로세스 내에서 프로그램을 실행하는 작업 단위이다.
- 프로세스 내부에서 프로세스의 일부 기능을 갖고 있기 때문에 경량프로세스(LWP, Light Weight Process)라고도 한다.
- 프로세스에는 하나 혹은 여러 개의 Thread를 포함한다.
- 작업을 수행할 때는 Thread가 반드시 필요하다.
- 하나의 Thread 는 독립된 작업수행과 작업제어를 하기 위한 자기만의 스택과 레지스터를 갖는다.

1.2 스레드의 형상화



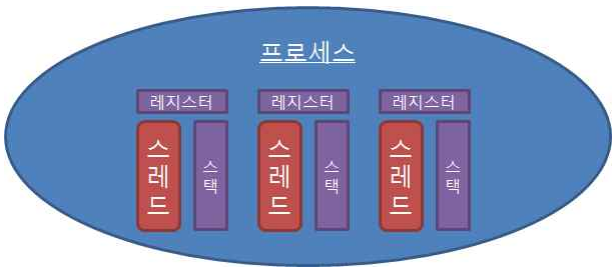
소주제2

2. 스레드의 종류

2.1 프로세스 사용방법에 따른 분류

(1) 단일 Thread
- 하나의 프로세스에 한 개의 Thread 존재 경우

(2) 다중 Thread
- 하나의 프로세스에 여러 개의 Thread 존재 경우



2.2 운영체제의 종류에 따른 분류

(1) 사용자 수준의 Thread
- 사용자가 만든 라이브러리를 사용하여 Thread 운용한다.
- 사용자 수준 Thread 여러 개가 커널 Thread 하나로 매핑 된다.(다대일 스레드 매핑)
- 속도는 빠르지만 구현이 어렵다.
- 문맥 교환이 적다.
- 독자적 알고리즘이 필요하다.
- 대형시스템에 적당하다.
- CPU사용을 해제 못하면 시스템이 중단된다.

(2) 커널 수준의 Thread
- 운영체제의 커널에 의해서 Thread를 운용한다.

- 사용자 수준 스레드 한 개가 커널 Thread 하나로 매핑 된다.(일대일 스레드 매핑)
- 구현이 쉽지만 속도가 느리다.
- 윈도우NT/XP/2000, 리눅스, 솔라리스9 이상버전, OS/2, Mach. 등등
- 문맥 교환이 많다.
- 독자적 알고리즘이 필요없다.
- 대형시스템에 적당하지 않다.
- CPU사용을 해제 못하면 운영체제가 지원한다.

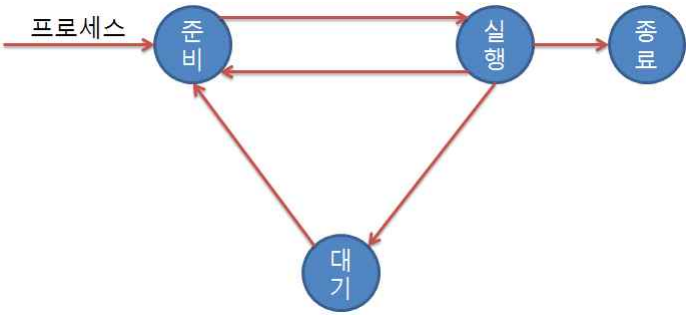
(3) 혼합형 Thread
- 사용자 수준의 스레드와 커널 수준의 스레드를 혼합한 형태로 운용된다.
- 다대다 스레드 매핑
- 솔라리스9 미만 버전

소주제3

3. 스레드의 상태

3.1 스레드의 상태

- 하나의 CPU가 하나의 스레드를 실행한다.
- 여러 개의 CPU가 여러 개의 스레드를 각각 동시에 처리하면 병렬처리이다.



① 준비

- 스레드가 프로세스에 의해서 실행될 수 있는 상태

② 실행

- 스레드가 프로세스에 의해서 실행 중인 상태 활성화 된 상태

③ 대기

- Thread가 입출력 작업이 완료될 때까지 대기 상태
- Thread 가 정보를 스택에 저장
- 같은 프로세스내의 다른 Thread 가 실행될 수 있다.

④ 종료

- Thread 가 작업을 완전히 종료한 상태
- 레지스터와 스택을 비운다.

소주제4

4. 스레드의 장점

4.1 스레드의 장점

1) 단일 프로세스를 여러 개의 스레드로 생성하여 병행성을 증진시킬 수 있다.
2) 하드웨어와 운영체제의 성능과 응용프로그램의 처리율을 향상시킬 수 있다.
3) 실행환경을 공유시켜서 기억장소의 낭비가 줄어든다.
4) 프로세스간의 통신 속도가 향상된다.
5) 공통적으로 접근 가능한 기억장치를 통해 효율적으로 통신한다.
6) 동일한 프로세스 환경에서 각각의 독립적인 다중 수행이 가능하다.
7) 스레드 기반 시스템에서 스레드는 독립적인 스케줄링의 최소단위로써 프로세스의 역할을 담당한다.

4주차 1차시

○ 들어가기

학습목표

- 스케줄링의 개념을 알고 설명할 수 있다.
- 스케줄링의 목적을 서술할 수 있다.
- 스케줄링의 분류방법에 따라 나누어 구분할 수 있다

주요용어

- 스케줄링 - 프로세스가 작업을 수행하는데 필요한 CPU공간을 할당 받기 위한 작업
- 응답시간 - 반응시간
- 반환시간 - 일정한 작업공간에서 작업이 완료되어 끝나는 시간
- 운휴상태 - CPU가 프로세스 진행이 없이 잠시 쉬는 상태

소주제1

1. 스케줄링의 개요

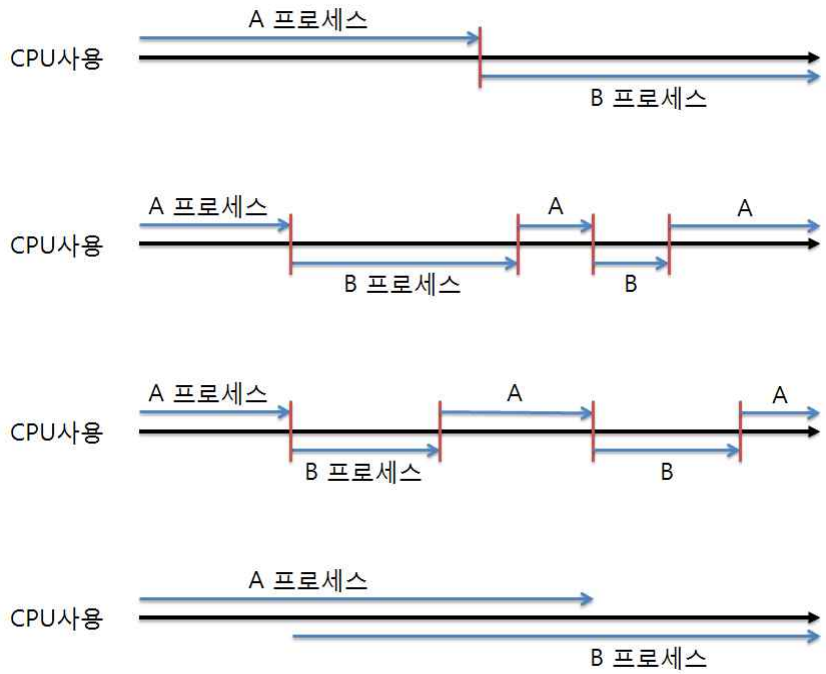
1.1 스케줄링(Scheduling)

프로세스가 작업을 수행하는데 필요한 CPU공간을 할당 받기 위한 작업이다.

1.2 프로세스 스케줄링

- CPU 스케줄링이라고도 한다
- CPU의 공간을 프로세스에게 배당하는 작업
- 하나의 프로세스는 완료시점까지 많은 스케줄링의 과정을 거친다.

1.3 스케줄링의 개념



소주제2

2. 스케줄링의 목적

2.1 스케줄링의 목적

① 공정성

프로세스들이 모두 공정하게 CPU를 할당한다.

② 처리량 증가

가능한 한 단위시간당 처리량(Through put)을 최대화 한다.

③ 응답시간 단축

대화식 사용자를 위한 프로세스 처리 후 반응시간을 최소화한다.

④ 반환시간 예측가능

시스템의 부하에 관계없이 일정한 작업량은 같은 시간 동안 같은 비용으로 실행, 완료되어야 한다.

⑤ 대기시간 단축

프로세스가 준비상태 큐에서 대기하는 시간을 최소화 한다.

⑥ CPU 이용률 증가

프로세스가 주기억장치를 사용하거나 또는 입출력명령에 의해 잠시 쉬는 상태(운휴상태)를 줄이고, CPU가 프로세스만 실행하도록 해야 한다.

⑦ 균형 있는 자원 활용

주기억장치와 입출력장치 등 시스템 내의 자원들을 골고루 사용한다.

⑧ 응답시간과 자원 활용간의 조화

- 응답시간을 빠르게 하려면 자원들을 충분히 확보하여 필요할 때마다 이용이 가능하도록 한다.
- 자원이 충분 할 경우 자원 활용도는 떨어진다.
- 실시간 시스템에서는 빠른 응답이 필수라서 자원 활용이 덜 중요하다.
- 빠른 응답시간이 아닌 정확도를 원하는 시스템에서는 효과적인 자원 활용이 더 중요하다.
- 시스템 응답시간과 자원 활용도 사이에서 적절한 조화를 이룬다.

⑨ 무한연기 배제

프로세스가 특정한 자원을 사용하기 위하여 무한정 연기되는 경우가 없어야 한다.

⑩ 우선순위 실시

모든 프로세서에 우선순위를 부여하여, 우선순위가 높은 프로세스를 먼저 실행한다.

⑪ 서비스 사용 가능

페이지 부재를 적게 발생시키는 프로세스에게 서비스 사용기회를 부여한다.

소주제3

3. 스케줄링의 분류

3.1 스케줄링의 종류

- 장기 스케줄링
  - 작업스케줄링(Job Scheduling), 상위 스케줄링
  - 작업 스케줄러에 의해서 수행
  - 어떤 프로세스가 시스템의 자원을 차지하게 할 것인가를 결정
  - CPU가 아닌 시스템 내의 자원들을 관리
- 중기 스케줄링
  - 어떤 프로세스들이 CPU를 할당 받을 것인지 결정하는 작업
  - CPU 스케줄링
- 단기 스케줄링
  - 프로세서 스케줄링(Processor Scheduling), 하위 스케줄링
  - 프로세서 스케줄러에 의해서 수행
  - 프로세스가 실행되기 위해 CPU를 할당 받는 시기와 특정 프로세스를 지정하는 작업

3.2 프로세스 스케줄링의 방법에 의한 분류

- 비선점 스케줄링
- 선점 스케줄링

3.3 프로세스 스케줄링 알고리즘에 의한 분류

(비선점 스케줄링)

- 우선순위 스케줄링
- 기한부 스케줄링
- FIFO 스케줄링
- SJF(Shortest Job First) 스케줄링
- HRN(Highest Response Ratio Next) 스케줄링

(선점 스케줄링)

- RR(Round Robin) 스케줄링
- SRT(Shortest Remaining Time) 스케줄링
- MFQ(Multilevel Feedback Queue) 스케줄링
- MLQ(Multi Level Queue) 스케줄링

4주차 2차시

○ 들어가기

학습목표

- 비선점 스케줄링의 개념을 정의할 수 있다.
- 알고리즘 기법에 따라 다른 비선점 스케줄링 종류를 구분할 수 있다.
- 비선점 스케줄링 내에서 에이징 기법의 사용을 학습하여 적용할 수 있다.

주요용어

- CPU 할당 - CPU를 사용 할 수 있도록 허가를 받아서 프로세스가 진행 중
- 비선점 - 프로세스 진행과정에서 CPU를 할당받아 사용하려고 우선순위로 미리 예약하지 않음

소주제1

1. 비선점(Non-Preemptive) 스케줄링의 개요

1.1 비선점 스케줄링의 개요

- 이미 CPU 공간이 할당되어 진행 중인 프로세스를 다른 프로세스가 빼앗아 사용 할 수 없는 스케줄링 기법이다.
- 모든 프로세스를 순차적으로 처리한다.
- 모든 프로세스의 요구를 공정하게 처리한다.
- 프로세스 응답시간 예측이 부정확하다.
- 중요한 짧은 작업이 중요하지 않은 긴 작업시간을 오랫동안 기다려야 하는 불편함도 있을 수 있다.
- 프로세스를 순차적으로 처리하기 때문에 일괄처리에 적합하다.
- FCFS, HRN, SJF, 기한부, 우선순위 등의 알고리즘 기법이 있다.

소주제2

2. 비선점 스케줄링의 종류(FCFS, HRN, SJF, 기한부, 우선순위 알고리즘)

2.1 FCFS(First Come First Service, 선입선출) 스케줄링

- FIFO(First In First Out)이라고도 한다.
- 준비상태 큐에 도착하는 순서대로 차례대로 CPU를 할당하는 기법
- 가장 간단한 알고리즘
- 먼저 도착한 작업이 먼저 처리되는 공정성
- 짧은 작업이 긴 작업을 기다리는 불편함

▷ FIFO(First In First Out)

프로세스	A	B	C
실행시간	10	5	15

- 평균 실행시간 : (10+5+15)/3=10
- 평균 대기시간 : (0+10+15)/3=8.3
- 평균 반환시간 : (10+15+30)/3=18.3

2.2 SJF(Shortest Job First, 최단시간 작업 우선) 스케줄링

- SJN(Shortest Job Next)이라고도 한다.
- 준비상태 큐에서 기다리는 프로세스 중에 실행시간이 가장 짧은 프로세스에게 먼저 CPU를 할당하는 기법
- 가장 적은 평균대기시간을 제공하는 최적 알고리즘 기법
- 짧은 작업시간을 갖는 프로세스에게 유리하다.
- 긴 작업시간을 갖는 프로세스는 짧은 작업시간을 갖는 프로세스에게 우선순위에서 계속 밀리면 무한 연기 될 수 있다.

▷ SJN(Shortest Job Next)

프로세스	A	B	C
실행시간	10	5	15

- 평균 실행시간 : (5+10+15)/3=10
- 평균 대기시간 : (0+5+15)/3=6.7
- 평균 반환시간 : (5+15+30)/3=16.7

2.3 HRN(Highest Response-ratio Next) 스케줄링

- 실행시간이 길어서 상대적으로 대기시간도 길었던 SJF 기법을 보완한 기법
- 실행시간과 대기시간을 이용한 우선순위 공식을 이용하여 우선 순위를 정한다.
- 실행시간이 짧거나 대기시간이 긴 프로세스에게 우선순위가 높다.
- 우선순위를 계산한 값이 큰 수부터 우선한다.

▷ 우선순위 계산식

* $우선순위 = \frac{(대기시간 + 실행시간)}{실행시간}$

프로세스	A	B	C
실행시간	10	5	15
대기시간	5	10	10
우선순위 계산	(10+5)/10=1.5	(5+5)/5=3	(15+10)/15=1.7

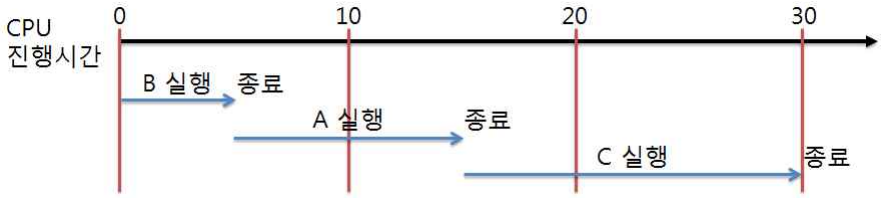
- 우선순위 : B → C → A

2.4 기한부(Deadline) 스케줄링

- 프로세스들이 일정한 시간 안에 프로세스가 완료되도록 하는 기법
- 프로세스가 일정한 시간 안에 완료되지 못하면 진행 중인 프로세스는 제거되거나 처음부터 다시 진행된다.
- 시스템은 작업시간 할당을 정확하게 해야 한다.
- 사용자는 시스템이 요구하는 프로세스 정보를 정확히 제시하여야 한다.
- 많은 기한부 작업들이 동시에 실행되면 스케줄링이 복잡해진다.
- 프로세스 실행 시 작업관리가 집중적으로 요구 되므로 오버헤드 발생한다.

▷ 기한부(Deadline) 스케줄링

프로세스	A	B	C
실행시간	10	5	15



- 실행순서 : B->A->C, B->C->A, A->B->C…
- 실행제거 : C->B->A, C->A->B, A->C->B…

2.5 우선순위(Priority) 스케줄링

- 준비상태 큐에 대기중인 프로세스에 우선순위를 부여하여 우선순위 순서대로 CPU를 할당하는 기법
- 우선순위가 같은 경우에는 FCFS기법 적용
- 우선순위가 낮을 경우 무한정 기다려야 한다는 단점이 있다.

▷ 우선순위(Priority) 스케줄링

프로세스	A	B	C
실행시간	10	5	15
우선순위	2	3	1

- 실행순서 : C → A → B

소주제3

3. 에이징(Aging) 기법

3.1 에이징 기법

- 기아상태를 해결하기 위한 기법으로써 오랫동안 기다린 프로세스에게 높은 우선순위를 부여하여 처리하는 기법이다.
- 에이징 기법은 프로세스의 우선순위가 낮아서 무한정 기다리는 상태를 방지하기 위하여 기다린 시간에 비례해서 일정 시간이 지나면 우선순위를 부여함으로써 우선순위가 낮은 프로세스를 처리하는 방법이다.
- SJF 기법에서 상대적으로 실행시간이 긴 프로세스들이 기나긴 오랜시간 동안 CPU를 할당받지 못하는 상황을 예방하기 위해서 에이징 기법을 도입해 HRN(HRRN) 기법으로 사용한다.

4주차 3차시

○ 들어가기

학습목표

• 선점 스케줄링의 개념을 정의할 수 있다.
• 알고리즘 기법에 따라 다른 선점 스케줄링의 종류를 구분할 수 있다.
• 비선점 스케줄링 내에서 에이징 기법의 사용을 적용할 수 있다.

주요용어

• 선점 - 프로세스 진행과정에서 CPU를 할당받아 사용하려고 우선순위로 미리 예약
• 큐(Queue) - FIFO(First In First Out)형태로 진행되는 레지스터의 한 기법

소주제1

1. 선점 스케줄링의 개요

1.1 선점 스케줄링의 개요

- 이미 CPU 공간이 할당되어 진행 중인 프로세스를 우선순위가 높은 다른 프로세스가 빼앗아 사용 할 수 있는 스케줄링 기법이다.
- 우선순위가 높은 프로세스를 먼저 빨리 처리 할 수 있다.
- 빠른 응답시간을 요구하는 대화식 시분할 시스템이나 실시간 처리에 효과적이다.
- 우선순위에 의한 선점 때문에 오버헤드를 초래한다.
- 일정시간 할당하도록 인터럽트용 타이머 클록(Clock) 필요하다
- RR, SRT, 다단계 큐(MQ), 다단계 피드백 큐(MFQ), 선점 우선순위 등의 알고리즘 기법 있다.

소주제2

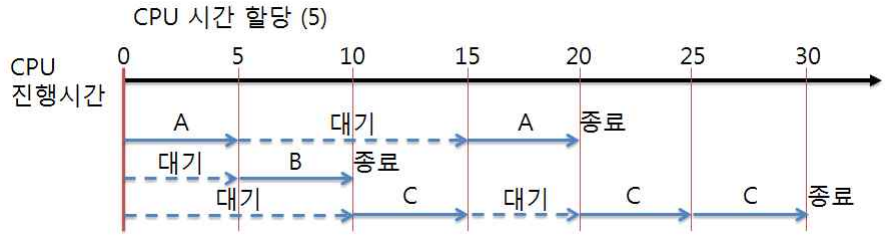
2. 선점 스케줄링의 종류(RR, SRT, 다단계 큐, 다단계 피드백 큐, 선점 우선순위 알고리즘)

2.1 RR(Round Robin) 스케줄링

- FCFS 기법을 선점 형태로 변형한 기법
- 시분할 시스템에서 효과적으로 사용된다.
- 시분할에 의한 동작을 하므로 할당하는 시간의 크기에 따라 CPU를 사용하는 효과에 대한 영향이 크다.
- 할당시간이 크면 FCFS와 같은 동작을 한다.
- 할당시간이 작으면 잦은 문맥교환으로 인한 오버헤드가 자주 발생한다.
- 준비상태 큐에 도착하는 순서의 차례대로 CPU를 할당 받아 실행하지만 할당된 일정시간 동안 실행을 완료하지 못 했을 경우, 다음 순위의 프로세스에게 CPU를 비워주고 준비상태 큐의 맨 뒤로 들어간다.
- 준비상태 큐에 도착하는 순서의 차례대로 CPU를 할당 받아 실행하는 기법은 FCFS하고 같다.

▷ RR(Round Robin) 스케줄링

프로세스	A	B	C
실행시간	10	5	15



프로세스	A	B	C	평균
반환시간	20	10	30	60/3=20
대기시간	15-5=10	5	25-10=15	30/3=10

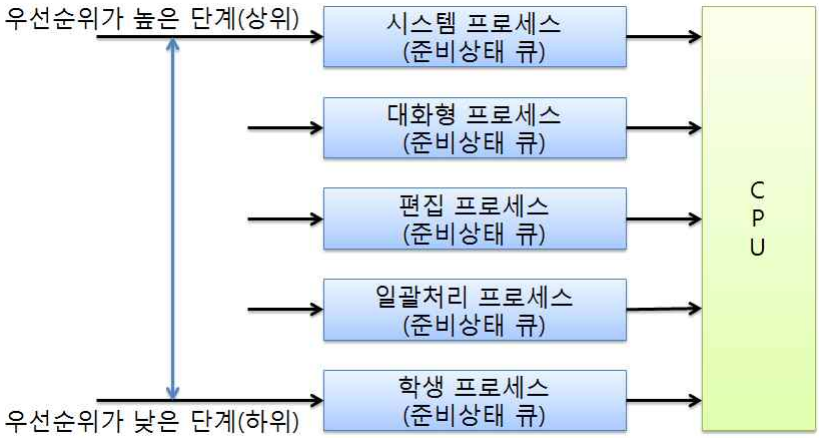
2.2 SRT(Shortest Remaining Time) 스케줄링

- 선점 SJF 기법이라고도 한다.
- 비선점 SJF기법을 선점 형태로 변형한 기법이다.
- 현재 실행중인 프로세스의 남은 실행시간과 준비상태 큐에 새로 들어온 프로세스의 실행시간을 비교하여 더 짧은 시행시간을 갖는 프로세스에게 CPU를 할당하는 기법이다.
- 시분할 시스템에 효과적이다
- 실행 중인 프로세스와 준비상태 큐에 있는 프로세스의 실행시간을 추적하여 보유하고 있어야 하므로 오버헤드 발생이 증가한다.
- 긴 작업을 요구하는 프로세스는 SJF보다 대기시간 길다.

2.3 다단계 큐(MQ, Multi-level Queue) 스케줄링

- 프로세스를 여러 개의 그룹으로 나눠서 각각의 그룹마다 각각의 준비상태 큐를 사용하는 기법
- 시스템 프로세스, 대화형 프로세스, 편집 프로세스, 일괄처리 프로세스 등으로 나누고 각 프로세스 별로 준비상태 큐를 배치한다.
- 각각의 준비상태 큐는 독자적인 스케줄링 기법 사용한다.
- 프로세스가 각각의 준비상태 큐 가운데 한군데 진입하면 다른 준비상태 큐로 이동할 수 없다.
- 우선순위가 높은 단계의 준비상태 큐에 프로세스가 들어오면 낮은 단계의 준비상태 큐에 있는 프로세스는 CPU를 비워주고, 우선순위가 높은 단계의 준비상태 큐에 있는 프로세스가 CPU를 할당받는다.

▷ 다단계 큐(MQ, Multi-level Queue) 스케줄링



2.4 다단계 피드백 큐(MFQ, Multi-level Feedback Queue) 스케줄링

- 프로세스를 여러 개의 그룹으로 나눠서 각각의 그룹마다 각각의 준비상태 큐를 사용하는 기법
- 프로세스가 각각의 준비상태 큐 가운데 한군데 진입하더라도 다른 준비상태 큐로 이동할 수 있다.
- 각 준비상태 큐 별로 시간 할당량을 주고 할당시간 안에 프로세스를 완료하지 못하면 다음 준비상태 큐 단계로 이동한다.
- 상위단계 준비상태 큐가 우선순위가 높으면서 시간 할당량이 적다.
- 우선순위가 높은 단계의 준비상태 큐에 프로세스가 들어오면 낮은 단계의 준비상태 큐에 있는 프로세스는 CPU를 비워주고, 우선순위가 높은 단계의 준비상태 큐에 있는 프로세스가 CPU를 할당받는다.
- 마지막 단계 준비상태 큐에서는 작업이 완료될 때까지 RR 스케줄링 기법 사용한다.

① 우선순위 결정

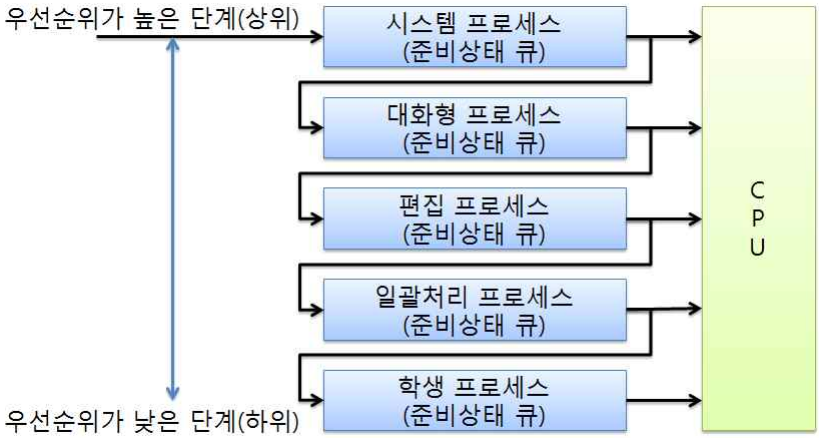
- 요구 시간이 적은 프로세스
- 입출력 중심의 프로세스
- 낮은 우선순위에서 오래 기다린 프로세스

② MFQ 스케줄링 알고리즘을 정의하는 요소

- 큐의 수
- 각각의 큐에 대한 스케줄링 알고리즘
- 작업을 높은 우선순위의 큐로 격상시키는 시기를 결정하는 방법

- 작업을 낮은 우선순위의 큐로 격하시키는 시기를 결정하는 방법
- 프로세스들이 어느 큐에 들어 갈 것인가를 결정하는 방법
- 프로세스가 서비스를 받는 시기를 결정하는 방법

▷ 다단계 피드백 큐(MFQ, Multi-level Feedback Queue) 스케줄링



2.5 선점 우선순위 스케줄링

- 준비상태 큐의 프로세스 중에 우선순위가 가장 높은 프로세스에게 CPU를 할당 하는 기법
- 비선점 스케줄링 기법 중에 우선순위 스케줄링을 선점 우선순위 스케줄링 기법 형태로 변경한 것이다.
- 준비상태 큐에 새로 들어오는 프로세스가 현재의 프로세스보다 우선순위가 높을 경우 현재의 프로세스를 보류하고 새로운 프로세스를 실행하는 기법

소주제3

3. 에이징(Aging) 기법

3.1 에이징 기법

- 기아상태를 해결하기 위한 기법으로써 오랫동안 기다린 프로세스에게 높은 우선순위를 부여하여 처리하는 기법이다.
- 에이징 기법은 프로세스의 우선순위가 낮아서 무한정 기다리는 상태를 방지하기 위하여 기다린 시간에 비례해서 일정 시간이 지나면 우선순위를 부여함으로써 우선순위가 낮은 프로세스를 처리하는 방법이다.
- MQ 기법에서 높은 우선순위를 먼저 처리하거나 다른 방법으로 우선순위를 높이는 방법으로 프로세스를 처리하기 위해서 에이징 기법을 적용하여 MFQ 기법으로 사용한다.

5주차 1차시

○ 들어가기

학습목표

- 병행프로세스를 학습하여 서술할 수 있다.
- 임계구역을 설명할 수 있다.
- 상호배제의 유형을 구분하여 설명할 수 있다.
- 동기화 기법을 학습하여 서술할 수 있다.

주요용어

- 임계구역-하나의 프로세스가 자원을 사용 중인 경우 그 프로세스만 그 자원을 사용하도록 지정해주는 영역
- 상호배제-어떤 프로세스가 자원을 사용 중인 경우 다른 프로세스가 그 자원을 사용할 수 없도록 하는 기법
- 세마포어-프로세스에 대한 제어 신호를 전달하여 순서대로 작업이 진행도록 하는 기법
- 모니터-공유자원을 프로세스에게 할당하기 위한 데이터와 데이터를 처리하는 프로시저로 구성된 동기화를 구현하기 위한 기법

소주제1

1. 병행 프로세서

1.1 병행 프로세서

- 두 개 이상의 프로세스들이 동시에 실행 상태를 말한다.
- 자원을 공유해서 컴퓨터 시스템에서의 프로세서가 하는 작업을 동시에 하기 위함이다.

① 독립적 병행 프로세스 : 프로세스들이 독립적으로 실행
② 협동적 병행 프로세스 : 프로세스들이 서로 협력하며 실행

- 다중프로그램, 다중처리, 분산처리에서 매우 중요하게 사용하는 개념이다.

소주제2

2. 임계구역

2.1 임계구역

- 다중 프로그래밍에서 많이 사용하는 개념으로써 여러 개의 프로세스가 여러 개의 데이터 및 공유자원을 가지고 실행 중일 때 어느 하나의 프로세스가 하나의 자원 혹은 데이터를 실행 중인 경우 오직 그 프로세스에서만 사용하도록 지정해 주는 자원이나 영역이다.
- 하나의 프로세스가 사용 중인 자원을 또 다른 프로세스가 사용 할 수 없다.
- 어떤 프로세스가 임계구역에 들어가면 다른 모든 프로세스는 임계구역으로의 진입이 금지 된다.
- 특정한 프로세스가 임계구역을 독점하지는 못한다.
- 특정 프로세스만을 위한 것이 아니므로 독점할 수 없다.
- 임계구역은 여러 프로세스가 사용해야 하므로 임계구역 내에서의 작업은 빠르게 이루어져야 한다.
- 현재 임계구역에서 실행되고 있는 프로세스가 없다면 잔류영역에서 임계구역으로의 진입을 기다리는 프로세스의 사용을 허락해야 한다.

소주제3

3. 상호배제

◇ 어떤 프로세스가 공유자원을 한 번에 하나만 사용할 수 있을 경우 다른 프로세스가 같은 공유자원을 사용할 수 없도록 하는 기법이다.

- 여러 프로세스가 사용하는 공유자원에 대해서는 각 프로세스가 공유자원을 차례로 하나씩 사용하도록 해야 한다.(임계구역 유지기법)

◇ 프로세스들이 서로 경쟁하면서 공유자원으로의 접근을 시도하면서 서로 경쟁하고 인식하며 프로세스간 경쟁과 협력이 나타난다.

- 프로세스간 서로 정보교환이 이루어지지 않으면 서로의 수행에 영향을 미친다.
- 서로 경쟁관계에 있는 프로세스들은 상호배제가 필요 하다.

3.1 소프트웨어적 구현

◇ 두 개의 프로세스 기준

- 데커(Dekker) 알고리즘
  - 네덜란드의 수학자 테오도루스 데커가 상호배제를 위해 고안한 병행 프로그래밍 알고리즘이다.
  - 두 개의 프로세스를 위한 상호배제의 최초의 소프트웨어 해결법이다.

- 피터슨(Peterson) 알고리즘
  - 수학자 개리 피터슨(Gary Peterson) 1981년에 로체스터 대학에서 발표
  - 상호배제를 위한 병렬 프로그래밍 알고리즘

◇ 여러 개의 프로세스 기준

- 램포트(Lamport)의 제과점 알고리즘 (Bakery Algorithm)
  - 분산처리 환경에 유용한 알고리즘이다.
  - 빵집에서 번호표를 가지고 빵을 사는 것처럼 준비상태 큐에서 대기 중인 프로세스에 우선순위를 부여

- 다익스트라(Dijkstra)의 N-process 상호배제
  - 여러 개(n개)의 프로세스 간의 상호배제 문제를 소프트웨어적으로 해결한 최초의 알고리즘
  - 실행시간이 가장 짧은 프로세스에 우선순위를 부여하여 프로세서 할당

3.2 하드웨어적 구현

- Test And Set 기법
- Swap 명령어 기법
  - Swap명령의 정의 : 두 워드의 내용을 교체하는 하드웨어 명령이다.

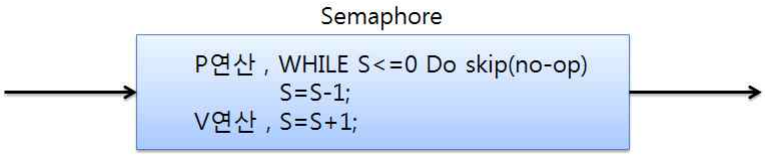
소주제4

4. 동기화 기법

- 여러 개의 프로세스를 동시에 실행을 할 수 없으므로 각 프로세스에 대한 처리 순서를 결정하는 것이다.
- 여러 개의 프로세스가 어떤 자원을 사용할 때 다른 프로세스가 같은 자원을 동시에 사용하지 못하도록 실행을 제어하는 기법으로 상호배제의 한 형태
- 순차적으로 재사용 가능한 자원을 공유하기 위한 프로세스 처리 순서를 결정하는 기법

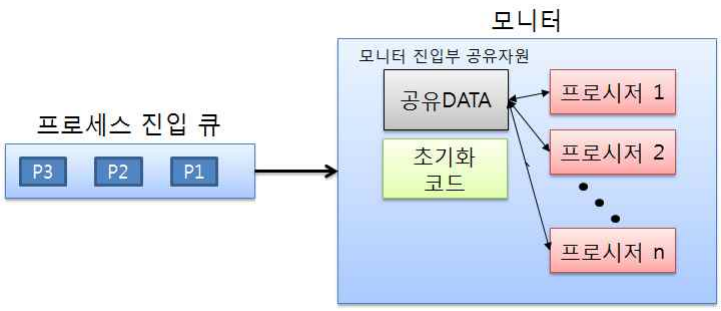
4.1 세마포어(Semaphore)

- 기차 선로의 차단기, “신호기”, “깃발”
- 프로세스에 대한 제어 신호를 전달하여 순서대로 작업이 진행되는 기법
- P와 V 두 개의 연산으로 동기화 유지
  - P : 프로세스를 대기 시키는 Wait 동작
  - V : 대기중인 프로세스를 깨우는 신호를 보내는 Signal 동작
- S, P와 V로만 접근 가능한 세마포어 변수



4.2 모니터(Monitor)

- 공유자원을 프로세스에게 할당하기 위한 데이터와 데이터를 처리하는 프로시저로 구성된 동기화를 구현하기 위한 기법
- 세마포어 보다 제어가 쉽다.
- 자료 추상화와 정보 은폐 개념을 기초로 한다.
- 모니터 내의 공유자원을 사용하기 위해서는 모니터의 반드시 진입부를 호출해야 한다.
- 외부 프로시저의 직접 엑세스 불가
- 모니터는 한순간에 하나의 프로세스만 진입하여 활동하도록 보장
- Wait 연산, Signal 연산



5주차 2차시

○ 들어가기

학습목표

- 교착상태의 개요를 설명할 수 있다.
- 교착상태의 발생 요인을 분석할 수 있다.
- 교착상태의 발생 조건을 활용할 수 있다.

주요용어

- 교착상태-자원을 점유하고 있는 두 개의 프로세스가 서로 다른 상대의 자원을 요구하면서 무한정 기다리는 상태
- 자원-CPU, 주기억장치, 보조기억장치, 디스크, 테이프, 프린터, 스캐너, 디바이스, 버스, 파일, 데이터베이스, 인터럽트, 신호, 메시지, 입출력 버퍼 같은 입출력 장치를 비롯한 주변장치와 통신 신호

소주제1

1. 교착상태의 개요

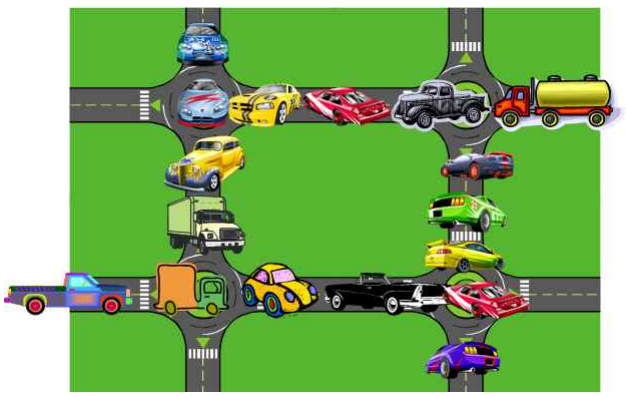
1.1 교착상태란?

- 두 개 이상의 프로세스가 서로 자원을 점유한 상태에서 자기에게 필요한 또 다른 자원을 서로 다른 프로세스가 갖고 있는 자원을 요구하며 무한정 기다리는 상태를 의미한다.
- 또 다른 표현은 하나 또는 여러 개의 프로세스들은 프로세스를 진행 중에 또 다른 자원을 요구하며, 일어나지 않을 사건(자원 할당)을 무한정 기다리는 상태를 말하기도 한다.
- 보통은 다중프로그래밍 상태에서 발생한다.

1.2 교착상태의 정의

- 하나 또는 둘 이상의 프로세스가 더 이상 계속할 수 없는 어떤 특정 사건(자원 할당)을 기다리는 상태를 말한다.
- 둘 이상의 서로 다른 프로세스가 요구한 자원을 할당 받아 점유하고 있으면서 상호간에 상대방 프로세스가 가진 자원을 요구하는 때를 말한다.

[교통의 교착상태]



소주제2

2. 교착상태의 발생

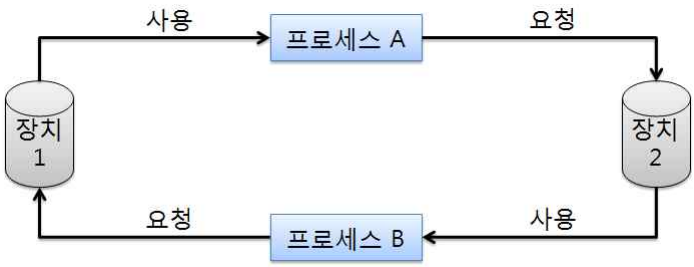
2.1 파일을 요청할 때

- 두 개의 프로세스와 두 개의 파일
- 파일을 이용해서 작업이 실행되는 동안, 또 다른 파일에 대한 다른 작업의 요청이 이루어지면 교착상태 발생

2.2 전용장치를 할당할 때

- 두 개의 프로세스가 각각 장치(드라이브)를 한 대씩 사용하면서, 서로 상대의 장치를 사용 요청을 할 경우 교착상태 발생

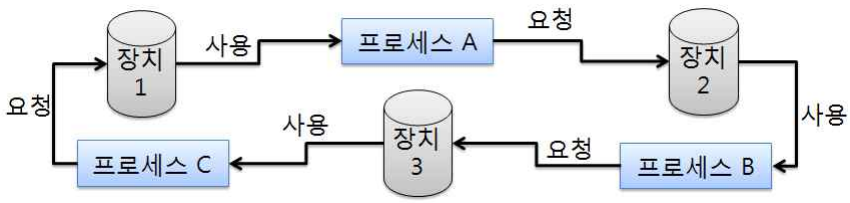
▷ 교착상태 발생 - 파일요청, 전용장치 할당



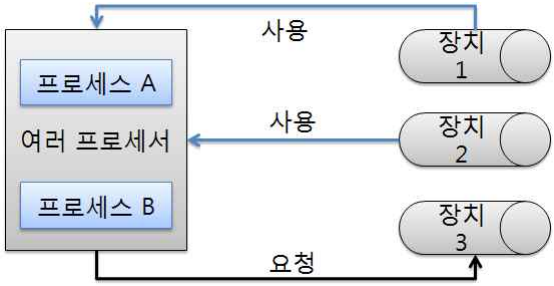
2.3 여러 주변장치를 할당할 때

- 여러 개의 프로세스가 전용 주변장치들을 요구할 때
- 세 개의 프로세스가 세 개의 전용장치를 사용하는 경우에 교착상태 발생

▷ 교착상태 발생 - 세 개의 전용장치 할당



▷ 교착상태 발생 - 여러 주변장치 할당



2.4 스풀링 시스템에서 출력할 때

- 스풀링 시스템이 교착상태에 빠지기 쉽다.
- 디스크에 할당된 스풀링 공간의 출력이 완료되지 않은 상태에서 다른 작업이 스풀링 공간을 모두 차지하게 되면 교착상태 발생
- 교착상태 발생 줄이기

- 스풀링 공간을 많이 할당 -> 비용 증가
- 스풀링 작업의 일정량 제한 설정 ->처리량 감소

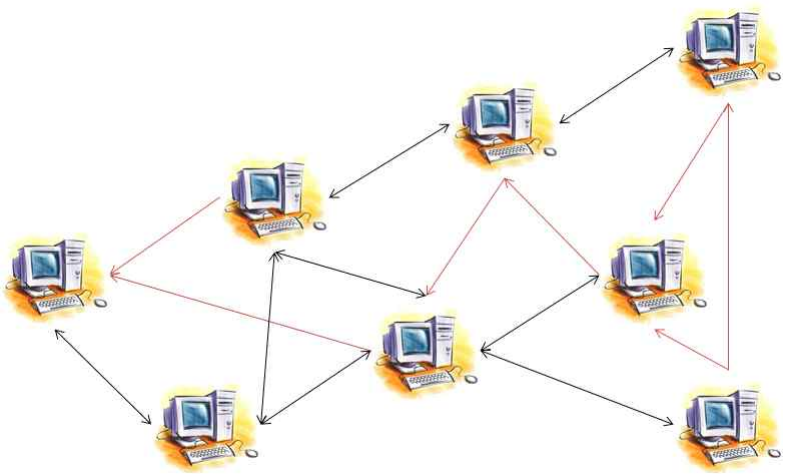
2.5 디스크를 공유할 때

- 디스크 사용에 대한 제어가 없을 경우 교착상태 발생
- 두 개의 프로세스가 두 개의 실린더 혹은 두 개의 섹터로 접근하여 서로 각각의 자료를 요구할 경우 교착상태 발생

2.6 네트워크에서의 교착상태

- 네트워크의 흐름을 제어하는 프로토콜이 없을 경우 교착상태 발생
- 네트워크의 교통량증가, 혹은 입출력 버퍼공간 부족할 경우 교착상태 발생

▷ 교착상태 발생 - 네트워크



소주제3

3. 교착상태의 발생 조건 (필요 충분 조건)

다음의 4가지 조건 중에 한 가지라도 충족하지 않으면 교착상태가 발생하지 않는다.
즉, 4가지 조건을 모두 만족하는 경우(필요 충분 조건)에 교착상태가 발생할 가능성이 있다.

(1) 상호배제
- 한 번에 하나의 프로세스만 해당자원을 사용할 수 있어야 한다.

(2) 점유와 대기
- 최소한 하나의 자원을 가지고 있으면서 다른 프로세스에 할당된 자원을 갖기 위해서 대기하는 프로세스가 있어야 한다.

(3) 비선점
- 다른 프로세스에 할당된 자원은 강제로 뺏을 수 없다.

(4) 순환대기(환형대기)
- 교착상태 발생 때 반드시 요구되는 조건이다. 공유자원과 공유자원을 사용하기 위한 프로세스들이 원형으로 구성되어 있고, 자기에게 할당된 자원을 가지면서 앞 혹은 뒤의 프로세스가 가진 자원을 요청하면서 대기한다.

5주차 3차시

○ 들어가기

학습목표

- 교착상태 예방과 회피를 정의할 수 있다.
- 교착상태 탐지와 회복을 정의할 수 있다.
- 기아상태를 학습하여 설명할 수 있다.
- 에이징 기법을 서술할 수 있다.

주요용어

- 환형대기 - 모든 자원에 대하여 일련의 순서대로 고유번호를 부여하고 앞이나 뒤 어느 한 방향으로만 자원을 요구
- 기아상태 - 프로세스가 사용할 수 없는 자원을 계속 기다리면 교착상태의 발생 우려가 있어서, 교착상태를 예방하기 위하여 자원을 할당했을 때 발생되는 기다림의 결과

소주제1

1. 교착생태 예방과 회피

1.1 교착상태 예방기법

- 교착상태가 발생하지 않도록 시스템을 제어하는 방법이다.
- 다음의 네 가지 조건 중 하나를 제거 또는 부정함으로써 교착상태 예방이 수행된다.
  (1) 상호배제(Mutual Exclusion) 부정
  (2) 점유와 대기(Hold and Wait) 부정
  (3) 비선점(Non-Preemption) 부정
  (4) 환형대기(Circular Wait) 부정

(1) 상호배제(Mutual Exclusion) 부정
- 상호배제는 자원을 공유하지 않는 조건을 전제로 한다.
- 상호배제 부정은 한 번에 여러 개의 프로세스가 공유자원을 사용할 수 있도록 한다.

(2) 점유와 대기(Hold and Wait) 부정
- 프로세스 대기를 없애려면 프로세스가 실행되기 전에 필요한 모든 자원을 할당한다.
- 프로세스가 자원을 점유하지 않은 상태에서만 자원을 요청할 수 있도록 한다.

(3) 비선점(Non-Preemption) 부정
- 비선점을 부정하는 전제 조건은 이미 할당된 자원에 대해서 선점권을 갖지 않는다.
- 프로세스가 어떤 다른 자원을 요구할 때, 요청한 자원을 사용 가능한지 검사하여, 사용 가능하다면 점유하고 있는 자원을 반납하고 요구한 자원을 사용하기 위하여 대기한다.

(4) 환형대기(Circular Wait) 부정
- 모든 자원에 대하여 일련의 순서대로 고유번호를 부여한다.
- 각 프로세스는 현재 점유 중인 자원의 고유번호를 기준으로 앞이나 뒤 어느 한 방향으로만 자원을 요구한다.

1.2 교착상태 회피기법

- 교착상태가 발생하지 않는다는 가능성을 두지 않고, 교착상태가 발생하면 피해 나가는 방법이다.

▷ 교착상태 회피방법 2가지
① 프로세스의 시작거부
  - 프로세스의 요구로 인해 교착상태가 발생할 수 있다면 프로세스를 시작하지 않는다.
② 자원할당 거부
  - 프로세스의 요구로 자원을 할당하여 교착상태가 발생할 수 있다면 요청한 자원을 할당하지 않는다.

◇ 은행가 알고리즘(Banker`s Algorithm)

- 교착상태 회피기법에 주로 사용되는 알고리즘
- 다익스트라 (E. J. Dijkstra)가 제안
- 은행의 모든 고객의 요구가 충족되도록 현금을 할당하는 것과 동일한 기법이다.
- 안전상태란 각 프로세스에게 자원을 할당하여 교착상태가 발생하지 않고 완료되는 상태이다.
- 불안전상태란 교착상태가 발생하는 상태이다.
- 적용조건으로는 자원의 양과 프로세스의 수가 일정해야 한다.
- 프로세스의 모든 요구를 유한시간 내에 할당해야 한다.
- 사용자가 자원의 최대 필요량을 시스템에 미리 알려야 한다.

소주제2

2. 교착생태의 탐지와 회복

- 교착상태 발생을 예방하지 못했다면 교착상태가 발생했는지 검사하는 알고리즘 필요하다.
- 교착상태가 발생하였다면 이로부터 회복하기 위한 알고리즘도 필요하다.

2.1 교착상태 탐지기법

- 교착상태가 발생했는지 검사하여 교착상태에 있는 프로세스와 자원을 찾아내는 것
  - 교착상태 발견 알고리즘, 자원할당 그래프 사용
- 쇼사니(Shoshani)와 코프만(Coffman)이 탐지 알고리즘을 제안

2.2 교착상태 회복기법

(1) 프로세스의 종료

- 교착상태의 모든 프로세스를 중지시킨다.
- 교착상태의 프로세스를 하나씩 중지시킨다.
- 프로세스를 중지시키는데 있어서 최소비용으로 중지시키기 위한 방법으로 희생자(프로세스)를 선택한다.
  - 희생자 선정 시 고려사항
    - 프로세스의 남은 작업 완료시간.
    - 프로세스들의 우선순위
    - 희생시킬 프로세스의 개수
    - 프로세스가 사용하는 자원의 형태와 개수

(2) 자원 선점

- 교착상태의 프로세스가 점유 중인 자원을 선점하여 다른 프로세스에게 할당한다.
- 자원을 선점하여 다른 프로세스에게 할당하고 난 후 교착상태의 프로세스를 정지시킨다.
- 프로세스를 정지시킨 후 재시작한다.
- 우선순위가 낮거나, 수행된 경력이 적거나, 사용되는 자원이 적은 프로세스 등을 위주로 해당 프로세스의 자원을 선점한다.

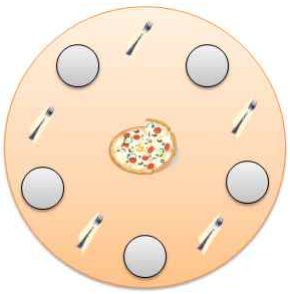
소주제3

3. 기아상태

3.1 기아상태

- 프로세스가 사용할 수 없는 자원을 계속 기다리면 교착상태의 발생 우려가 있어서, 교착상태를 예방하기 위하여 자원을 할당했을 때 발생되는 기다림의 결과이다.

▷ 식사하는 철학자 문제



▷ 해결방법

- 대기 중인 프로세스를 발견하고 각 프로세스가 대기한 시간을 조사하고 추적한다.
- 새로운 기아상태를 발견하면 새로운 작업의 시작을 보류한다.
  - 빈번한 시스템 보류로 인하여 처리량이 감소할 수 있다.

소주제4

4. 에이징(Aging) 기법

4.1 에이징 기법

- 기아상태를 해결하기 위한 기법으로써 오랫동안 기다린 프로세스에게 높은 우선순위를 부여하여 처리하는 기법이다.
- 에이징 기법은 프로세스의 우선순위가 낮아서 무한정 기다리는 상태를 방지하기 위하여 기다린 시간에 비례해서 일정 시간이 지나면 우선순위를 부여함으로써 우선순위가 낮은 프로세스를 처리하는 방법이다.
- 우선순위가 낮더라도 실행 순위를 기다리고 있을 수 있으니 늦게라도 자원이 할당되어 처리되도록 한다.
  - 대기중인 프로세스에 나이(age)를 부여한다.
  - 자원을 할당 받지 못하고 기다리는 동안 나이가 증가한다.
  - 나이를 우선순위에 반영한다.
  - 나이가 증가함에 따라 우선순위가 증가한다.

6주차 1차시

○ 들어가기

학습목표

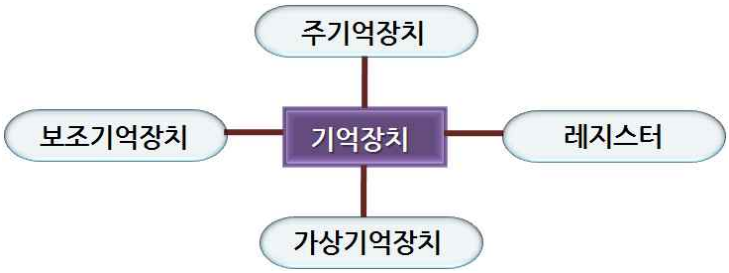
- 기억장치의 개요를 서술할 수 있다.
- 여러 기억장치의 구조와 특성에 대해 구분하여 설명할 수 있다.
- 여러 기억장치의 관리를 학습하여 기억장치 관리에 활용할 수 있다.

주요용어

- 캐시기억장치 - 주기억장치와 CPU사이에 존재하는 기억장치
- 레지스터 - 컴퓨터 내부시스템의 장치와 장치 사이에 존재하는 임시기억장소

소주제1

1. 기억장치의 개요



[기억장치 개요]

주기억장치, 보조기억장치, 캐시기억장치, 가상기억장치, 레지스터 등으로 이루어졌으며
컴퓨터 시스템에서 프로세스를 진행할 때 원시 DATA와 결과 DATA를 기억하고 저장하며 프로세스 진행에 필수 자원이다.

소주제2

2. 기억장치의 구조 및 특성

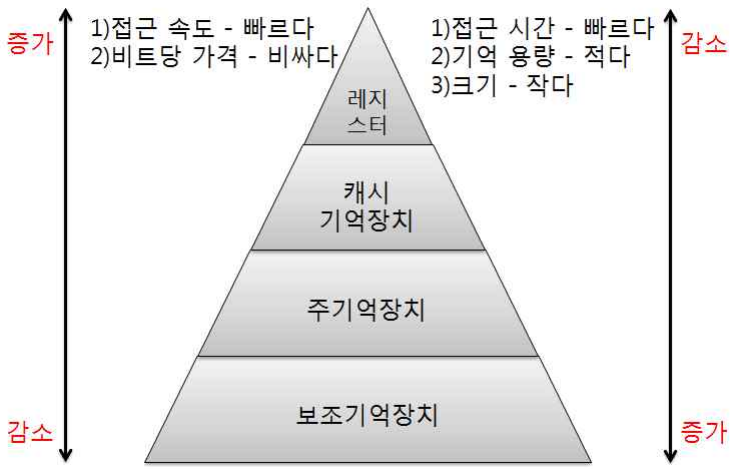
▷ 기억장치는 크게 주기억장치, 보조기억장치, 캐시기억장치, 레지스터 등으로 나누어진다.

- 주기억 장치- RAM, ROM
- 보조기억장치 - HDD, CD, DVD, USB, TAPE, DRUM CD
- 캐시기억장치 - 주기억장치와 CPU사이에 존재하는 기억장치
- 레지스터 - 컴퓨터 내부시스템의 장치와 장치 사이에 존재하는 임시기억장소

2.1 기억장치 계층구조의 특징

- 보통 레지스터를 상위구조로 보고 보조기억장치를 하위구조로 구분한다.
- 상위구조 장치일수록 접근속도가 빨라지면서 접근시간 또한 빨라진다.
- 상위구조 장치일수록 크기와 기억용량이 줄어들고 가격이 비싸다.
- 주기억장치는 워드, 블록 단위로 구성되며 각 단위마다 주소를 갖고 주소를 이용하여 접근한다.
- CPU가 직접 접근 가능한 장치는 보조기억장치를 제외한 주기억장치, 캐시기억장치, 레지스터 등이다
- 보조기억장치의 DATA를 CPU가 접근하기 위해서는 주기억장치로 옮겨진 후 접근 가능하다

2.2 기억장치의 계층구조



[기억장치 계층구조]

소주제3

3. 기억장치의 관리

3.1 기억장치의 관리 전략

- 주기억장치에서 사용하는 프로그램이나 DATA를 보조기억장치로 부터 옮겨오는 방법
- 보조기억장치로부터 주기억장치로의 적재시기와 적재위치를 정한다.
- 보조기억장치로부터 주기억장치로 옮겨온 프로그램이나 DATA를 효율적으로 공간을 활용하기 위하여 반입전략, 배치전략, 교체전략 이용한다.

(1) 반입(Fetch) 전략

→ 프로그램이나 DATA를 보조기억장치로부터 주기억장치로 적재할 시기를 결정하는 전략이다.

① 요구반입(Demand Fetch)

다음에 실행할 프로세스를 참조할 시기, 즉 실행 중인 운영체제나 프로그램 등의 참조요구에 따라서 프로그램이나 DATA를 주기억장치에 적재하는 방법이다.

② 예상반입(Anticipatory Fetch)

실행 중인 프로그램에 의해 시스템의 요구를 예측하여 참조될 프로그램이나 DATA를 미리 기억장치에 적재하는 방법이다.

(2) 배치(Placement) 전략

→ 보조기억장치로부터 들어온 프로세스 혹은 프로그램이나 DATA를 주기억장치의 어느 위치에 저장할 것인가를 결정하는 전략이다.

① 최초 적합(First Fit)

프로그램이나 DATA를 사용할 수 있는 빈 공간 중에서 첫 번째 공간에 할당하는 방법이다.

② 최적 적합(Best Fit)

프로그램이나 DATA를 사용할 수 있는 빈 공간 중에서 가장 작은 공간에 할당하는 방법이다.

③ 최악 적합(Worst Fit)

프로그램이나 DATA를 사용할 수 있는 빈 공간 중에서 가장 큰 공간에 할당하는 방법이다.

(3) 교체(Replacement) 전략

→ 주기억장치의 모든 영역을 차지하고 사용 중인 프로그램이나 DATA를 새로운 프로그램이나 DATA로 재배치하고자 할 때 사용하는 전략

① 교체전략

[FIFO, OPT, LRU, LFU, NUR, SCR 등이 있음]

- FIFO(First In First Out)
- OPT(OPTimal replacement, 최적적합)
- LRU(Least Recently Used) - 가장 오랫동안 사용하지 않은 페이지 교체
- LFU(Least Frequently Used) - 사용빈도가 가장 적은 페이지 교체
- NUR(Not Used Recently) - 최근에 사용하지 않은 페이지 교체
- SCR(Second Chance Replacement) - 가장 오랫동안 주기억장치에 있던 페이지 중 자주 사용되는 페이

지의 교체를 방지하기 위함

6주차 2차시

○ 들어가기

학습목표

- 주기억장치의 할당의 개념을 서술할 수 있다.
- 단일 분할 할당 기법을 설명할 수 있다.
- 다중 분할 할당 기법을 설명할 수 있다.

주요용어

- 오버레이(Overlay) 기법 - 주기억장치의 공간이 부족하면 중첩하여 적재하고 여러 번 중첩이 가능하다.
- 스와핑(Swapping) 기법 - 하나의 프로그램을 주기억장치 전체를 할당하여 사용하다가 다른 새로운 프로그램으로 교체하는 기법

소주제1

1. 주기억장치 할당의 개념

1.1 주기억장치의 할당

프로세스 진행을 위한 프로그램이나 DATA를 주기억장치에 할당하기 위한 방법이다.

(1) 연속 할당기법

- 프로그램이나 DATA를 연속으로 할당하는 기법
- 단일 분할 할당 기법
- 다중 분할 할당 기법

(2) 분산 할당기법 (가상기억장치에서…)

- 프로그램이나 DATA를 조각으로 나누어 할당하는 기법
- 페이징(Paging) 기법
- 세그멘테이션 (Segmentation) 기법

소주제2

2. 단일 분할 할당 기법

한 번에 한 명의 사용자만이 주기억장치를 사용하는 기법이다.

- DOS 환경에서 사용하던 기법
- 사용자가 주기억장치에 관한 모든 제어 함
- 운영체제 상주를 위한 영역과 사용자를 위한 영역으로 나누는 경계 레지스터(Boundary Register)가 있음
- 사용자 프로그램이 작으면 기억장치의 사용자 영역이 낭비 됨
- 주기억장치보다 큰 프로그램은 원래 실행이 되지 않지만 오버레이(Overlay) 기법을 사용하여 해결 함

2.1 오버레이(Overlay) 기법

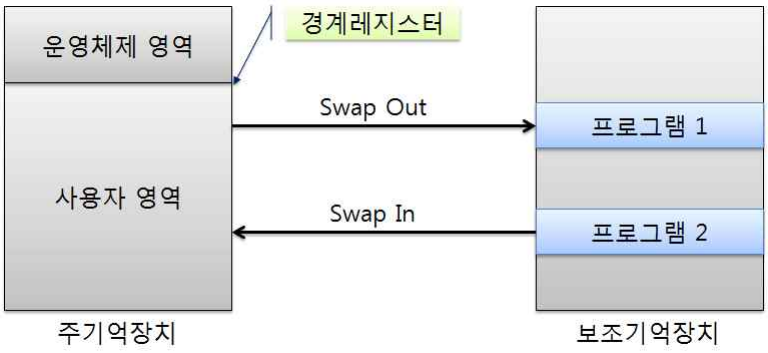
- 주기억장치 영역보다 큰 사용자 프로그램을 실행하기 위한 기법이다.
- 보조기억장치에 저장된 하나의 프로그램을 여러 개의 조각으로 분할한 후 주기억장치에 차례로 적재한다.
- 이때, 주기억장치의 공간이 부족하면 중첩하여 적재한다.
- 여러 번 중첩가능하다.

[오버레이(Overlay) 기법]

2.2 스와핑(Swapping) 기법

- 하나의 프로그램을 주기억장치 전체를 할당하여 사용하다가 다른 새로운 프로그램으로 교체하는 기법이다.
- Swap Out : 주기억장치의 프로그램이 보조기억장치로 이동한다.
- Swap In : 보조기억장치의 프로그램이 주기억장치로 이동한다.
- 여러 번 교체가능하다.

[스와핑(Swapping) 기법]

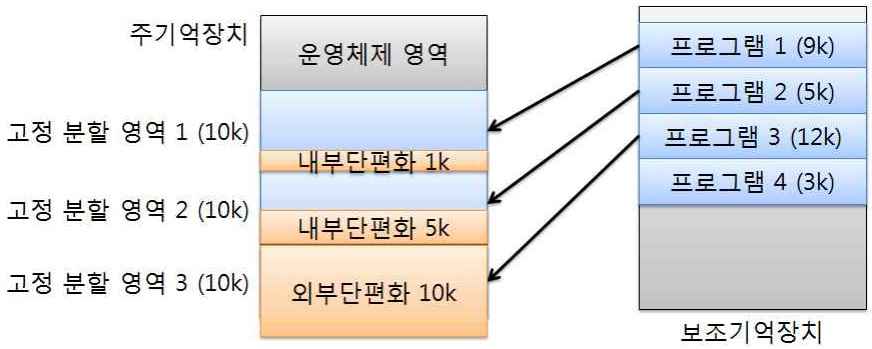

운영체제 영역
경계레지스터
Swap Out
프로그램 1
사용자 영역
Swap In
프로그램 2
주기억장치
보조기억장치

소주제3

3. 다중 분할 할당 기법

3.1 고정 분할 할당(Multiple contiguous Fixed parTition allocation, MFT) 기법

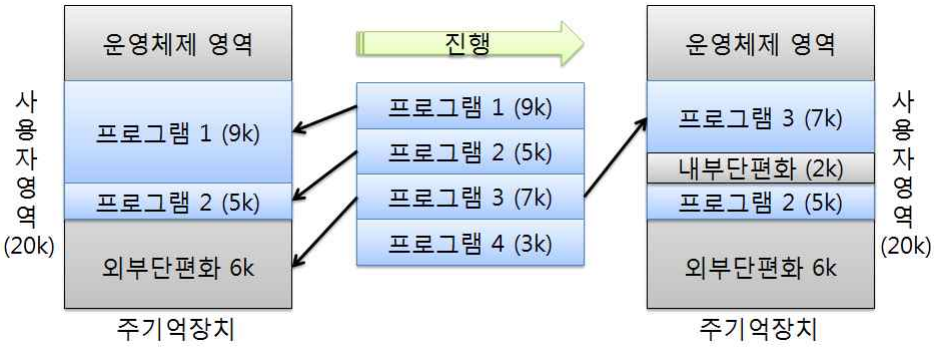
- 정적 할당(Static Allocation) 기법 이라고도 한다.
- 주기억장치의 사용자 영역을 고정 크기로 분할한 후 작업상태 큐의 프로그램을 분할된 영역에 할당하여 수행하는 기법이다.
- 프로그램 전체가 주기억장치에 있어야 한다.
- 프로그램이 고정 분할된 크기 보다 클 경우 할당되지 못하는 경우 있다.
- 분할된 크기가 모두 같아서 크기가 각기 다른 프로그램이 할당될 경우, 내부단편화, 외부단편화가 발생한다.
- 실행할 프로그램 크기를 미리 알아야 한다.



[고정 분할 할당기법]

3.2 가변 분할 할당(Multiple contiguous Variable parTition allocation, MVT) 기법

- 동적 할당(Dynamic Allocation) 기법 이라고도 한다.
- 고정 분할 할당 기법의 단편화를 줄이기 위한 방법이다.
- 프로그램을 주기억장치에 할당 하면서 그때그때 필요한 만큼의 크기로 영역을 분할하는 기법이다.
- 주기억장치를 효율적 사용도가 좋아진다.
- 다중프로그래밍의 실행 정도를 높인다.
- 할당되어 실행될 프로그램의 크기가 다양하다.
- 내부 or 외부 단편화는 생기지 않지만 영역과 영역 사이에 단편화가 생길 수 있다.



[가변 분할 할당기법]

- 프로그램 3이 처음에 할당되어 외부 단편화가 발생 된다.
- 프로그램 1이 실행되고 난 후, 빈 공간이 생기면, 프로그램 3을 프로그램 1이 사용되고 난 후 생긴 공간에 할당한다.
- 내부단편화가 발생한다.

6주차 3차시

○ 들어가기

학습목표

• 단편화 현상이 발생하여 주기억장치의 공간을 낭비하는 현상을 분석할 수 있다.
• 주기억장치의 낭비를 막기 위하여 단편화를 줄이거나, 병합하는 기법을 학습하여 활용할 수 있다.

주요용어

• 내부단편화(Internal Fragmentation)-분할된 영역이 프로그램이나 DATA의 크기보다 커서 사용되지 않고 남은 부분
• 외부단편화(External Fragmentation)-분할된 영역이 프로그램이나 DATA의 크기보다 작아서 사용되지 않고 남은 부분

소주제1

1. 단편화

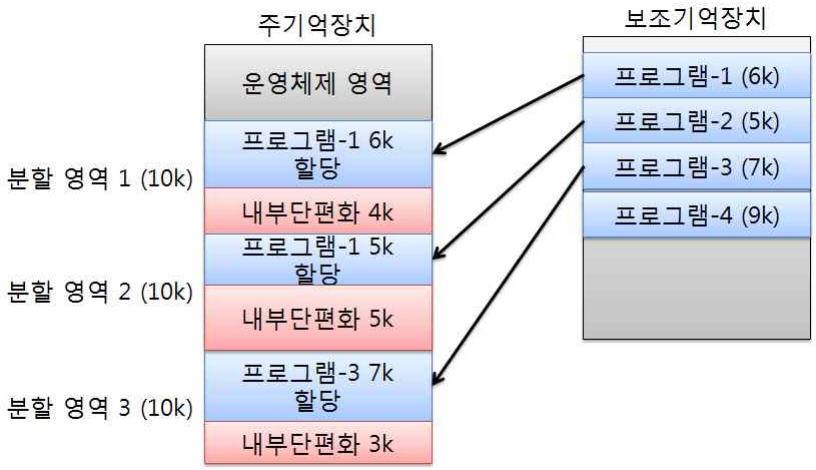
주기억장치를 분할하여 사용할 때 프로그램이나 DATA를 할당했을 때 사용되지 않고 남는 조각을 말한다.

- 페이징이나 세그먼트와 비슷하다.
  (페이징은 일정한 크기 · 세그먼트는 가변 크기)
- 내부단편화(Internal Fragmentation)
- 외부단편화(External Fragmentation)

1.1 내부단편화(Internal Fragmentation)

- 분할된 영역이 프로그램이나 DATA의 크기보다 커서 사용되지 않고 남은 부분을 말한다.
- 페이징에서 나타난다.
- 동일한 크기의 메모리 할당에서 발생한다.

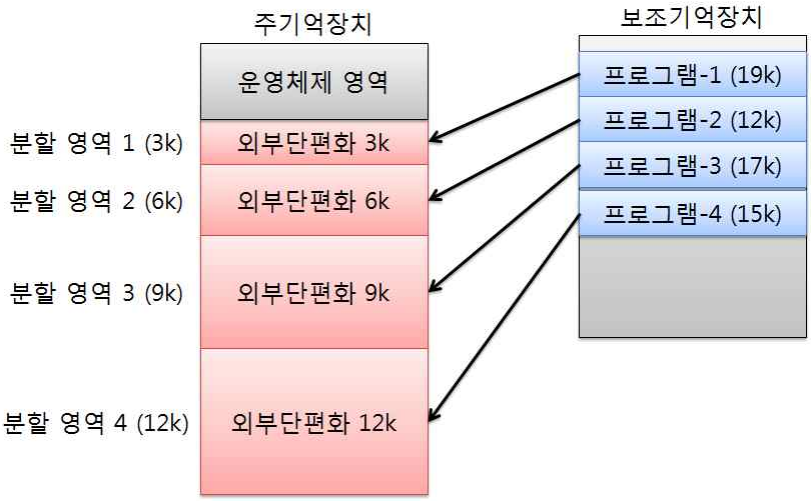
[내부단편화]



1.2 외부단편화(External Fragmentation)

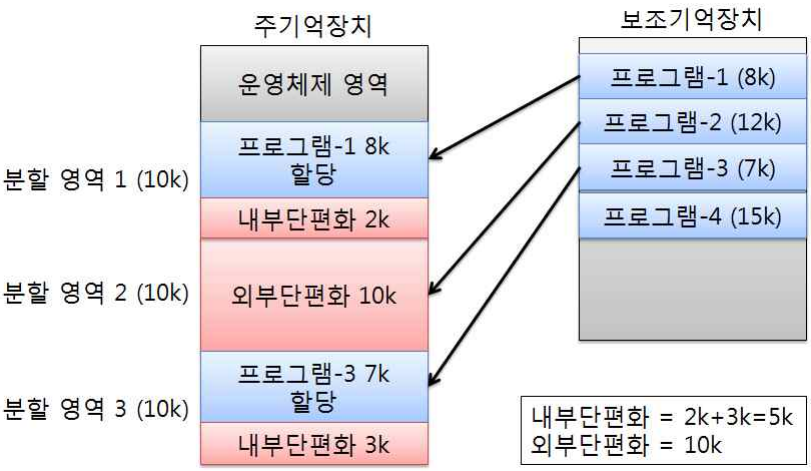
- 분할된 영역이 프로그램이나 DATA의 크기보다 작아서 사용되지 않고 남은 부분을 말한다.
- 세그멘테이션에서 나타난다.
- 가변크기의 메모리 할당에서 발생한다.
- 외부단편화가 발생하면 기다리거나 압축하여 더 큰 공간을 만든다.
- 평균 세그먼트의 크기가 작으면 외부단편화도 작다.

[외부단편화]



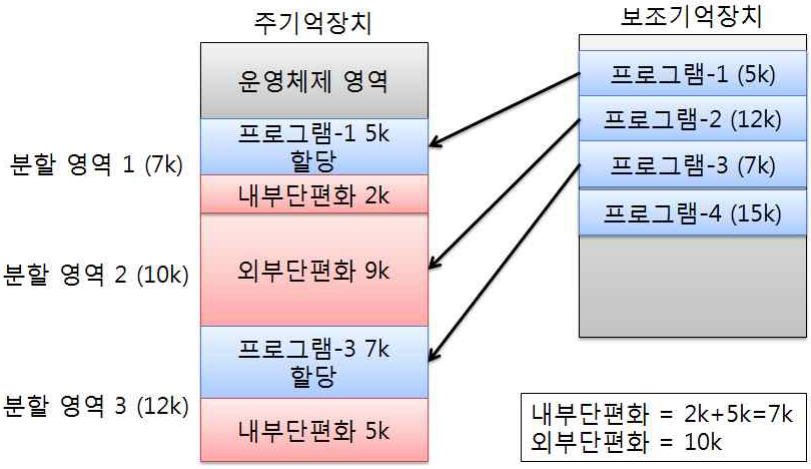
1.3 단편화 예시(1)

[고정 분할]



1.4 단편화 예시(2)

[가변 분할]



소주제2

2. 단편화 해결 방법

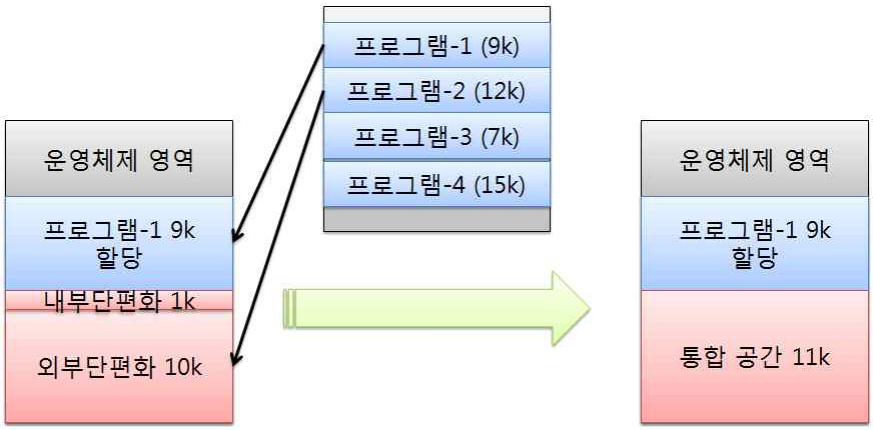
- 조각으로 나뉘어져 단편화 되어진 주기억장치의 공간을 모아서 커다란 하나의 공간으로 만든다.
- 통합기법과 압축기법이 있다.

▷ 시작 > 모든 프로그램 > 보조프로그램 > 시스템 도구 > 디스크 조각 모음

2.1 통합기법

- 조각으로 나뉘어져 단편화 되어진 주기억장치의 공간 중에 인접해 있는 단편화된 공간끼리 묶어서 통합하는 기법이다.
- 주기억장치에 단편화가 나타났을 때 두 개의 단편화 조각이 인접해 있는지를 먼저 조사해야 한다.

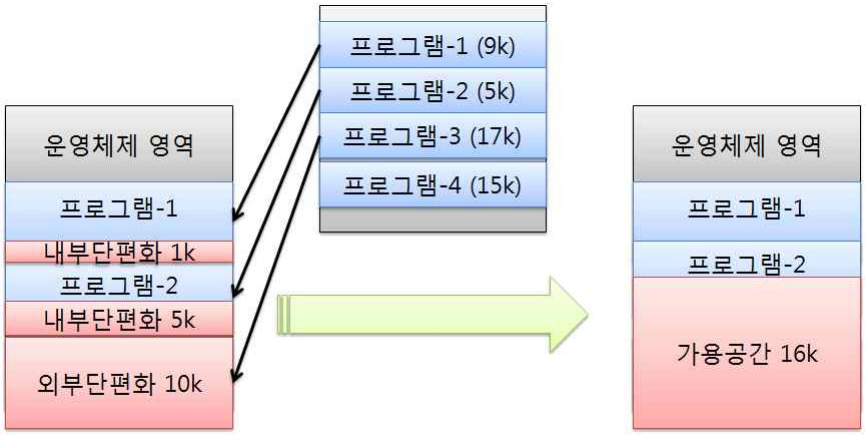
[통합 기법]



2.2 압축기법

- 조각으로 나뉘어져 단편화 되어진 주기억장치의 공간을 하나의 커다란 공간으로 묶어서 통합하는 기법
- **쓰레기 수집(Garbage Collection)**이라고도 한다.
- 주기억장치의 여러 곳에서 단편화된 조각을 모두 주기억장치의 끝으로 옮겨서 커다란 가용공간 생성한다.
- 압축기법이 진행되는 동안에는 시스템의 모든 작업이 “일시 중지” 된다.

[압축 기법]


프로그램-1 (9k)
프로그램-2 (5k)
프로그램-3 (17k)
프로그램-4 (15k)
운영체제 영역
프로그램-1
내부단편화 1k
프로그램-2
내부단편화 5k
외부단편화 10k
운영체제 영역
프로그램-1
프로그램-2
가용공간 16k

7주차 1차시

○ 들어가기

학습목표

- 가상기억장치의 개요를 설명할 수 있다.
- 가상기억장치의 구현방법을 서술할 수 있다.
- 가상기억장치의 관리전략을 유형별로 구분할 수 있다.

주요용어

- 프리페이징 - 처음부터 과도한 페이지 부재를 방지하기 위하여, 프로세스 실행에 필요할 것 같은 모든 페이지를 한꺼번에 페이지 프레임에 적재하는 기법.
- 스레싱(Thrashing) - 프로세스의 처리시간보다 페이지 교체시간이 더 많아지는 현상

소주제1

1. 가상기억장치의 개요

- 보조기억장치의 일부분을 주기억장치인 것처럼 사용하는 방법이다.
- 주기억장치의 기억용량이 작으므로 기억용량이 큰 보조기억장치의 저장 공간 이용한다.
- 주기억장치의 기억용량보다 큰 프로그램을 실행하기 위하여 사용한다.
- 가상기억장치에는 프로그램을 여러 개의 블록단위로 나누어서 보관한다.
- 주기억장치에서 실행요구가 발생하면 가상기억장치의 블록단위로 저장된 프로그램을 주기억장치에 보내서 실행한다.
- 주기억장치의 이용률 증가한다.
- 다중프로그래밍 효율 증가한다.
- 주기억장치와 가상기억장치의 주소가 다르므로 프로그램을 실행할 경우, 가상기억장치에서 주기억장치의 주소로 바꾸는 주소변환 작업 필요하다.
- 가상기억장치는 주소변환과정이 필요하므로 시스템 오버헤드가 발생할 수 있다.
- 블록단위로 나누어 사용하기 때문에 단편화를 해결할 수 있다.
- 가상기억장치는 구현이 어렵다.

소주제2

2. 가상기억장치의 구현방법

▷ 일반적인 구현방법으로는 블록 종류에 따라 페이징 기법과 세그먼테이션 기법이 있다.

2.1 페이징 기법

- 프로그램을 같은 크기로 나누어 블록으로 사용하는 기법이다.
- 프로그램을 같은 크기로 나눈 단위를 페이지라고 한다.

2.2 세그먼테이션 기법

- 프로그램을 가변적인 크기로 나누어 블록으로 사용하는 기법이다.
- 프로그램을 가변적인 크기로 나눈 단위를 세그먼트라고 한다.

소주제3

3. 가상기억장치의 관리전략

가상기억장치를 구현할 때 시스템 성능에 영향을 미치는 여러 가지 관리 사항이 있다.

① 페이지 크기
② Locality
③ 워킹 셋(Working Set)
④ 페이지 부재 빈도 방식
⑤ 프리페이징(Prepaging)
⑥ 스레싱(Thrashing)

2.1 페이지 크기

▷ 프로그램을 페이지 단위로 나누는 것을 말한다.

(1) 페이지 크기가 작을 경우
- 페이지 단편화 감소
- 페이지 하나가 주기억장치로 이동하는 시간 감소
- 프로세스 수행에 필요한 내용만 주기억장치에 적재
- Locality(국부성)에 더욱 일치 가능성 때문에 기억장치 효율 증가
- 페이지 Map 테이블 커지고, Mapping 속도 감소
- 디스크 접근횟수 및 I/O시간 증가

(2) 페이지 크기가 클 경우
- 주기억장치에 적재되는 프로그램 수 감소
- 프로세스 수행에 불필요한 내용까지 주기억장치에 적재 가능
- “페이지 크기가 작을 경우”와 반대 상황

2.2 Locality(국부성, 지역성, 구역성, 국소성)

▷ 프로세스가 실행되는 동안 주기억장치의 일부 페이지만 집중적으로 참조한다.
- 스레싱 방지를 위한 Working Set 이론의 기반이 되었음
- 가상기억장치 관리의 이론적인 근거
- Denning 교수에 의한 구역성 개념 증명
- 캐시 기억장치 시스템의 이론적 근거

(1) Locality의 종류

① 시간 구역성(Temporal Locality)
- 프로세스가 실행되면서 하나의 페이지를 일정한 시간 동안 집중적으로 액세스 함
- 한번 참조한 페이지는 빠른 시간 내 계속 참조 할 가능성이 높음을 의미
- 시간 구역성이 이루어지는 기억장소

- Loop, Stack, 부 프로그램, Counting, 집계에 사용되는 변수

② 공간 구역성(Spatial Locality)

- 프로세스 실행 시 일정한 위치의 페이지를 집중적으로 액세스 함
- 어느 하나의 페이지를 참조하면 근처의 페이지를 계속 참조 할 가능성이 높음을 의미
- 공간 구역성이 이루어지는 기억장소

- 배열 순회, 순차적 코드의 실행, 프로그램 변수 근처의 할당 장소, 같은 영역의 변수 참조

2.3 워킹 셋(Working Set)

▷ 프로세스가 일정시간 동안 자주 참조하는 페이지들의 집합이다.

- Denning이 제안, 프로그램의 Locality 특징을 이용
- 자주 참조되는 워킹셋을 주기억장치에 상주 시켜서 페이지 부재 및 페이지 교체 현상을 줄인다.
- 주기억장치 내의 워킹셋을 감소시키면 페이지 프레임 공간을 적게 차지하면서 워킹셋을 사용하므로 스레싱 감소 -> 주기억장치에 최소한의 워킹셋 올려놓아야 한다.
- 워킹셋 설정시간 간격이 증가하면 주기억장치에 유지되는 워킹셋은 커지고, 감소하면 작아진다.
- 오버헤드 발생 가능성 있다.

2.4 페이지 부재 빈도 방식

▷ 페이지부재(Page Fault) : 프로세스 실행 시 참조할 페이지가 주기억장치에 없는 현상

▷ 페이지부재빈도(Page Fault Frequency) : 페이지 부재가 일어나는 횟수

- 페이지 부재율에 따라 주기억장치에 있는 프레임 수를 늘리거나 줄여서 페이지 부재율을 적정수준으로 유지하는 방식
- 페이지 프레임을 할당하고, 페이지 부재율을 감시하다가 페이지 부재율이 상한선을 넘어가면 페이지 프레임을 할당하고, 하한선을 넘어가면 페이지 프레임을 회수한다.

2.5 프리페이징(Prepaging)

- 처음부터 과도한 페이지 부재를 방지하기 위하여, 프로세스 실행에 필요할 것 같은 모든 페이지를 한꺼번에 페이지 프레임에 적재하는 기법
- 페이지들 중에 사용되지 않는 페이지가 많을 수 있다.

2.6 스레싱(Thrashing)

▷ 프로세스의 처리시간보다 페이지 교체시간이 더 많아지는 현상이다.

- 다중프로그래밍, 가상기억장치 사용시스템에서 발생하며, 프로세스 진행 과정 중에 페이지 부재가 자주 발생 시 나타나는 현상으로, 시스템 전체 성능 저하
- 다중프로그래밍의 정도가 높아짐에 따라 CPU 활용도는 높아지지만 더욱 커지면 스레싱 발생하고, CPU 이용률은 급격히 감소
- CPU 이용률 높이고 스레싱 발생 방지하기 위하여, 다중프로그래밍 정도를 적정수준으로 유지하고, 페이지 부재빈도를 조절해서 사용하며, 워킹 셋을 유지해야 한다.

7주차 2차시

○ 들어가기

학습목표

- 페이징 기법을 학습하여 서술할 수 있다.
- 세그멘테이션 기법을 학습하여 설명할 수 있다.

주요용어

- 페이지(Page) - 프로그램을 일정한 크기로 나눈 단위
- 세그먼트(Segment) - 프로그램을 배열이나 함수등과 같은 논리적인 크기로 나눈 단위

소주제1

1. 페이징 기법

[개요]

▷ 가상기억장치의 프로그램 영역과 주기억장치의 영역크기를 동일크기(페이지)로 나눈 후, 가상기억장치의 페이지(프로그램 조각)를 동일하게 나누어진 주기억장치의 페이지 프레임(영역)에 적재하여 실행하는 기법이다.

▷ 외부단편화는 발생하지 않지만 내부 단편화가 발생한다.

① 페이지(Page)

- 프로그램을 일정한 크기로 나눈 단위

② 페이지 프레임(Page Frame)

- 페이지 크기로 일정하게 나누어진 주기억장치의 영역 단위

③ 페이지 맵 테이블(Page Map Table)

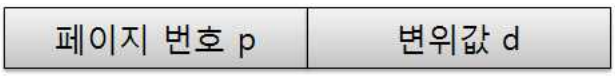
- 가상기억장치의 가상주소와 주기억장치의 실제주소 사이의 주소를 연결시켜주기 위하여 필요하다.(주소 매핑)
- 주소변환을 위한 페이지의 위치 정보를 가지고 있다.

1.1 페이징 기법의 주소 변환

① 가상주소 형식

- 페이지번호 : p
- 변위 값 : d

→ 페이지 내에서 실제 내용이 위치하는 곳까지의 거리



② 실기억주소 형식

- 페이지 프레임 번호 : pf
- 변위 값 : d

→ 페이지 프레임 내에서 실제 참조 위치까지의 거리

③ 페이지 맵 테이블(Page Map Table)

- 디스크 주소

→ 페이지가 주기억장치에 없을 때 보조기억장치의 주소

- 프레임 번호

→ 페이지가 주기억장치에 있을 때의 페이지 프레임 번호

- 상태비트

→ 사용할 페이지가 주기억장치에 존재하는지의 여부 표시

디스크 주소	페이지 프레임 번호	상태 비트

1.2 페이징 기법의 주소 변환 순서

소주제2

2. 세그멘테이션(Segmentation)기법

[개요]

▷ 가상기억장치에 보관된 프로그램을 다양한 크기의 논리적 단위로 나눈 후 주기억장치에 적재하여 실행하는 기법이다.

① 세그먼트(Segment)

- 프로그램을 배열이나 함수등과 같은 논리적인 크기로 나눈 단위
- 각 세그먼트는 고유한 이름과 크기를 갖는다.

② 기억장치의 사용자 관점을 보존하는 기억장치 관리 기법

③ 세그먼테이션 활용하는 궁극적인 이유

- 기억공간 절약

④ 세그먼트 맵 테이블(Segment Map Table)

- 주소변환을 위한 세그먼트가 존재하는 위치정보를 가지고 있다.

⑤ 기억장치 보호키(Storage Protection Key)

- 세그먼트가 주기억장치에 적재될 때 다른 세그먼트에게 할당된 영역을 침범 할 수가 없다.

2.1 세그먼테이션 기법의 주소 변환

① 가상주소 형식

- 세그먼트 번호 : s
- 변위 값 : d

→ 페이지 내에서 실제 내용이 위치하는 곳까지의 거리

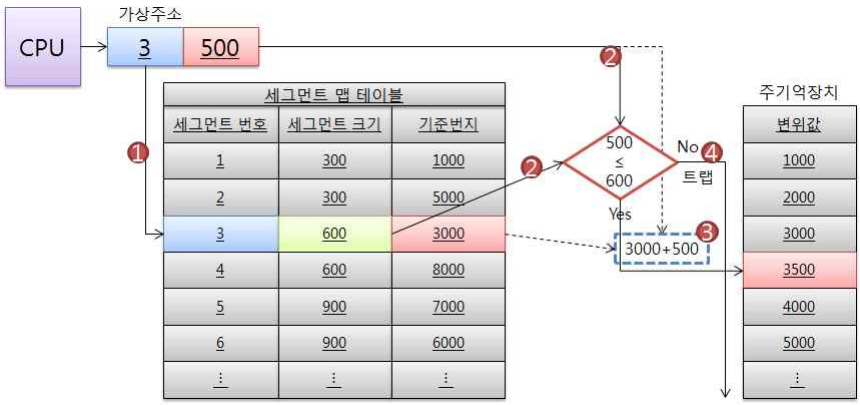
② 실기억주소 형식

- 완전주소 사용
- 세그먼트 기준번지 + 변위 값

③ 세그먼트 맵 테이블(Segment Map Table)

- 세그먼트 번호
- 세그먼트의 크기
- 주기억장치의 기준번지

2.2 세그먼테이션 기법의 주소 변환 순서



7주차 3차시

○ 들어가기

학습목표

- 페이지 교체 알고리즘의 종류를 구분할 수 있다.
- 페이지 교체 알고리즘의 기법을 학습하여 설명할 수 있다.

주요용어

- LFU(Least Frequently Used) - 사용빈도가 가장 적은 페이지 교체하는 기법
- LRU(Least Recently Used) - 가장 오랫동안 사용하지 않은 페이지 교체 기법

소주제1

1. 페이지교체 알고리즘의 종류

① FIFO(First In First Out, 선입선출)
- 각 페이지가 주기억장치에 가장 먼저 들어와서 가장 오래 있었던 페이지를 교체하는 기법

② LFU(Least Frequently Used, 최소빈도사용)
- 사용빈도가 가장 적은 페이지 교체하는 기법

③ LRU(Least Recently Used, 최근최소사용)
- 가장 오랫동안 사용하지 않은 페이지 교체 기법

④ NUR(Not Used Recently, 최근사용전무)
- 최근에 사용하지 않은 페이지 교체 기법

⑤ OPT(OPTimal replacement, 최적교체)
- 앞으로 가장 오랫동안 사용하지 않을 페이지를 교체하는 기법

⑥ SCR(Second Chance Replacement, 2차기회교체)
- 가장 오랫동안 주기억장치에 있던 페이지 중에서 자주 사용되는 페이지의 교체를 방지하기 위한 기법

소주제2

2. 페이지교체 알고리즘의 기법

2.1 FIFO(First In First Out, 선입선출)

▷ 각 페이지가 주기억장치에 가장 먼저 들어와서 가장 오래 있었던 페이지를 교체하는 기법이다.

- 이해가 간단하다.
- 프로그램 설계가 간단하다.
- 벨레이디 모순(Belady's Anomaly) 현상 발생

- 페이지 프레임 수가 증가함에도 페이지 부재가 더 많이 발생하는 현상

[FIFO 알고리즘 사용 예]

참조 페이지	7	0	1	2	3	2	0	3	4	1	5	2
페이지 프레임	7	7	7	2	2	2	2	2	4	4	4	2
		0	0	0	3	3	3	3	3	1	1	1
			1	1	1	1	0	0	0	0	5	5
페이지 부재	p	p	p	p	p		p		p	p	p	p

2.2 LFU(Least Frequently Used, 최소빈도사용)

▷ 사용빈도가 가장 적은 페이지 교체하는 기법이다.

- 활발하게 사용되는 페이지는 사용 횟수가 많아서 교체되지 않는다.
- 프로그램 초기에 사용 횟수가 많았던 페이지는 나중에 사용되지 않더라도 프레임을 계속 차지할 수 있다.

[LFU 알고리즘 사용 예]

참조 페이지	7	0	1	2	3	2	0	3	4	1	5	2
페이지 프레임	7	7	7	7	7	7	7	7	7	7	7	7
		0	0	0	0	0	0	0	0	0	0	0
			1	2	3	2	2	3	4	1	5	2
페이지 부재	p	p	p	p	p	p		p	p	p	p	p

2.3 LRU(Least Recently Used, 최근 최소 사용)

▷ 가장 오랫동안 사용하지 않은 페이지 교체 기법이다.

- 각 페이지마다 계수기 혹은 스택을 두고 페이지 사용시간 체크
- 각 페이지마다 계수기 혹은 스택 같은 별도의 하드웨어가 필요하다.
- 시간적 오버헤드가 발생
- 실제로 구현하기가 어렵다.

[LRU 알고리즘 사용 예]

참조 페이지	7	0	1	2	3	2	0	3	4	1	5	2
페이지 프레임	7	7	7	2	2	2	2	2	4	4	4	2
		0	0	0	3	3	3	3	3	3	5	1
			1	1	1	1	0	0	0	1	1	5
페이지 부재	p	p	p	p	p		p		p	p	p	p

2.4 NUR(Not Used Recently, 최근 사용전무)

- LRU 와 비슷한 알고리즘
- 최근에 사용하지 않은 페이지는 나중에도 사용하지 않을 가능성이 높다는 전제로 사용
- LRU에서 나타나는 시간적 오버헤드를 줄일 수 있다.
- 최근의 사용여부 체크를 위하여 각 페이지 마다 두 개의 비트 사용

- 참조비트(Reference Bit)
- 변형비트(Modified Bit, Dirty Bit)

[NUR 알고리즘 사용 예]

참조 비트	변형 비트	교체 순서
0	0	1
0	1	2
1	0	3
1	1	4

2.5 OPT(OPTimal replacement, 최적교체)

▷ 앞으로 가장 오랫동안 사용하지 않을 페이지를 교체하는 기법이다.

- 벨레이디(Belady)가 제안
- 페이지 부재 횟수가 가장 적게 발생하는 가장 효율적인 기법
- 각 페이지 호출 순서와 참조 상황을 미리 예측 하기 어렵다
- 실현가능성 매우 어려움

[OPT 알고리즘 사용 예]

참조 페이지	7	0	1	2	3	2	0	3	4	1	5	2
페이지 프레임	7	7	7	7	7	7	7	3	3	3	3	2
		0	0	0	0	0	0	0	4	4	4	4
			1	2	3	2	2	2	2	1	5	5
페이지 부재	p	p	p	p	p	p		p	p	p	p	p

2.6 SCR(Second Chance Replacement, 2차기회교체)

▷ 가장 오랫동안 주기억장치에 있던 페이지 중에서 자주 사용되는 페이지의 교체를 방지하기 위한 기법

- FIFO 기법의 단점 보완 기법이다.
- 각 페이지 별로 참조 비트를 두고

- FIFO기법을 이용하여 페이지 교체 수행
- 참조비트 0일 경우: 교체
- 참조비트 1일 경우: 참조 비트를 0으로 지정하고 FIFO 큐 리스트의 맨 마지막으로 보낸 후 다음 순서 대기

- 교체 대상이 되기 전에 참조비트(1)를 검사하여 한 번의 기회를 더 준다. (Second Chance)