흥달쌤과 함께하는

정보처리기사 실기

마무리 특강 학습교안

1억뷰 N잡

YouTube 흥달쌤 (https://bit.ly/3KtwdLG) E-Mail hungjik@naver.com

네이버 카페 흥달쌤의 IT 이야기 (https://cafe.naver.com/sosozl/)

01 운영체제

Section 1. 운영체제 기초

1. 기억장치

⑴ 기억장치의 개념

- 데이터, 프로그램, 연산의 중간 결과 등을 일시적 또는 영구적으로 저장하는 장치

⑵ 기억장치의 종류

1) 레지스터

- 중앙처리장치 내부에 존재하는 기억장치

2) 캐시 메모리

- 중앙처리장치가 주기억장치에 접근할 때 속도 차이를 줄이기 위해 사용

3) 주기억장치

- 중앙처리장치가 직접 데이터를 읽고 쓸 수 있는 장치
- 종류

종류	설명
ROM (Read Only Memory)	- 읽기만 가능한 읽기 전용 메모리 - 비휘발성 메모리 - 종류 : mask-ROM, PROM, EPROM, EEPROM
RAM (Random Access Memory)	- 기억장소를 임의로 접근할 수 있는 메모리 - 읽고 쓰기가 가능한 휘발성 메모리 - SRAM : 전원이 공급되는 중에 내용이 사라지지 않음(캐시 메모리로 사용) - DRAM : 일반적인 주기억장치로, 일정 시간이 지나면 내용이 사라지는 RAM

4) 보조기억장치

- 주기억장치에 비해 접근 시간은 느리지만 기억 용량이 크다.

5) 연관 메모리

- CPU가 찾고자하는 주기억장치 메모리가 캐시 메모리 어디에 있는지 빠르게 검색할 수 있게 사용하는 메모리
- 주소에 의해 접근하지 않고, 기억된 내용의 일부를 이용하여 Access할 수 있는 기억장치

2. 시스템 소프트웨어

⑴ 시스템 소프트웨어의 개념

- 응용 소프트웨어를 실행하기 위한 플랫폼을 제공
- 종류

종류	설명
로더	어떤 프로그램을 실행하기 위해 해당 목적 프로그램을 메모리에 적재하고 배치 주소를 옮기는 프로그램
링커	프로그램 구현 시 목적파일(Object File)을 실행파일(Execute File)로 변환해 주는 프로그램
유틸리티	컴퓨터 하드웨어, 운영체제, 응용 소프트웨어를 관리하는 데 도움을 주는 프로그램
번역기 (컴파일러, 어셈블러)	특정 프로그래밍 언어로 쓰여 있는 문서를 다른 프로그래밍 언어로 옮기는 프로그램
장치 드라이버	특정 하드웨어나 장치를 제어하기 위한 커널의 일부분으로 동작하는 프로그램
운영체제	CPU 메모리와 하드디스크 등의 하드웨어를 관리하며, 내 컴퓨터와 다른 컴퓨터들이 대화할 수 있도록 도와주는 등 많은 일들을 해주는 프로그램

⑵ 시스템 소프트웨어의 구성

1) 제어 프로그램

종류	설명
감시 프로그램 (Supervisor Program)	각종 프로그램의 실행과 시스템 전체의 작동 상태를 감시/감독하는 프로그램
작업관리 프로그램 (Job Control Program)	연속 처리를 위한 스케줄 및 시스템 자원 할당 등 담당
데이터 관리 프로그램 (Data Control Program)	주기억/보조기억장치 사이의 자료전송, 파일의 조작 및 처리, 입출력 자료와 프로그램 간의 논리적 연결 등 처리할 수 있도록 관리

2) 처리 프로그램

종류	설명
서비스 프로그램 (Service Program)	효율성을 위해 사용 빈도가 높은 프로그램
문제 프로그램 (Problem Program)	특정 업무 해결을 위해 사용자가 작성한 프로그램
언어 번역 프로그램 (Language Translator Program)	어셈블러, 컴파일러, 인터프리터

3. 운영체제

⑴ 운영체제의 개념

- 응용프로그램이 실행되는 과정에서 하드웨어들을 제어하여 응용프로그램을 실행시키고 실행 결과를 보일 수 있도록 컴퓨터 내부 동작을 관리하는 소프트웨어

⑵ 운영체제의 기능

기능	설명
프로세스 관리	프로세스를 생성하고 실행을 제어, 관리하는 기능
메모리 관리	프로세스가 실행될 수 있도록 메모리 공간을 할당하고 회수하는 기능
파일 관리	파일을 보조기억장치에 저장하고 파일 시스템을 운영하는 기능
입출력 관리	컴퓨터 시스템에서의 입력과 출력을 관리하는 기능
보조기억장치 관리	보조기억장치의 공간을 할당하고 관리하는 기능
네트워킹	컴퓨터 통신에 필요한 제어 관리 기능
정보 보안 관리	사용자 인증 및 실행 권한 관리
명령해석	사용자가 운영체제에 전달하는 명령을 해석하고 관련 함수를 실행시키는 기능

⑶ 운영체제 운용 기법

운용 기법	설명
일괄 처리 시스템 (Batch Processing System)	- 초기 운영체제의 형태로 여러 작업을 한 번에 묶어서 처리한다.
실시간 처리 시스템 (Real Time Processing)	- 요청한 실행을 즉시 실행하는 기법
다중 프로그래밍 시스템 (Multi Programming)	- 하나의 작업이 입출력 중일 때 다른 작업을 할당하여 CPU 사용률과 처리량을 향상시키는 기법 - 사용자 입장에서는 하나의 CPU이지만 동시에 여러 프로그램이 실행되는 것처럼 보인다.
시분할 시스템 (Time Sharing)	- 타임 슬라이스 또는 타임 퀀텀이라 부르는 일정 작업 시간 동안 작업 실행
다중 처리 시스템 (Multi-Processing)	- 여러 개의 CPU를 통하여 동시에 여러 개의 작업을 처리하는 운용 기법 - 병렬 처리 시스템(Parallel Processing System)이라고도 한다.
다중 모드 시스템 (Multi-Mode)	- 일괄 처리, 다중 프로그래밍, 시분할, 다중 처리, 실시간 처리시스템을 모두 혼용하여 사용
분산 처리 시스템 (Distribute Processing)	- 둘 이상의 독립된 시스템이 통신으로 연결되고 상호작용하는 약결합 방식

4. 운영체제의 종류

⑴ 윈도우(Windows)

- MS-DOS의 멀티태스킹 기능과 GUI 환경을 제공하는 운영체제

- 특징
  - GUI 제공
  - 선점형 멀티태스킹 방식
  - 자동감지 기능 제공 (Plug and Play)
  - OLE (Object Liking and Embedding) 사용

⑵ 리눅스(Linux)

- 1991년 리누스 토발즈에 의해 오픈소스로 개발된 유닉스 호환 OS
- 특징
  - 다중 사용자 시스템
  - 오픈 소스
  - 이식성/유연성/확장성
  - 다양한 배포판

⑶ 유닉스(Unix)

1) 유닉스 개요

- 1969년 미국 AT&T 벨 연구소(Bell Lab.)가 개발한 공개형 오픈소스 운영체제(OS)
- 1969년 벨연구소 소속의 켄 톰슨(Ken Tompson)이 어셈블리 언어를 사용하여 개발했으며, 1972년 데니스 리치(Dennis Ritchie)가 C언어를 사용하여 다시 작성

2) Unix 시스템의 구성

구성	설명
커널(Kernel)	- UNIX의 가장 핵심적인 부분 - 컴퓨터가 부팅될 때, 주기억장치에 적재된 후 상주하면서 실행
쉘(Shell)	- 명령어 해석기
유틸리티 프로그램 (Utility Program)	- 일반 사용자가 작성한 응용 프로그램을 처리하는 데 사용

4) Unix 파일 시스템

- Unix 파일 시스템 특징
  - UNIX 파일 시스템의 구조는 트리 구조로 되어 있다.
  - 디렉터리나 주변장치도 모두 파일로 취급
  - 파일 생성 및 삭제 기능, 보호 기능
  - 파일 형식은 일반파일, 디렉터리 파일, 특수파일 형태 제공
- Unix 파일 시스템의 구조

구조	설명
부트블록	부팅 시 필요한 코드를 저장하고 있는 블록
슈퍼블록	전체 파일 시스템에 대한 정보를 저장하고 있는 블록
I-node 블록	각 파일이나 디렉터리에 대한 모든 정보를 저장하고 있는 블록
데이터 블록	실제 파일에 대한 데이터가 저장된 블록

5) 파일 디스크립터(FD, File Descriptor)

- 파일 디스크립터 특징
  - 유닉스 시스템에서 프로세스가 파일들을 접근할 때 이용
  - 파일 제어 블록(File Control Block)이라고도 한다.
- 파일 디스크립터 정보
  - 파일 이름 및 파일 크기
  - 보조기억장치에서의 파일 위치
  - 파일 구조(순차파일/색인순차파일/색인파일 등)
  - 보조기억장치의 유형(자기 디스크/자기 테이프 등)
  - 액세스 제어 정보
  - 파일 유형(텍스트파일, 목적 프로그램 파일 등)
  - 생성 날짜와 시간, 제거 날짜와 시간
  - 최종 수정 날짜 및 시간
  - 액세스한 횟수(파일 사용 횟수)

6) POSIX(Portable Operating System Interface)

- 이식 가능한 운영체제 인터페이스
- 서로 다른 UNIX OS의 공통 API를 정리하여 이식성이 높은 유닉스 응용 프로그램을 개발하기 위한 목적으로 IEEE가 책정한 애플리케이션 인터페이스 규격

⑷ MacOS

- 애플사가 개발한 유닉스 기반의 운영체제
- iOS, tvOS, watchOS 등 애플의 다른 운영체제들과 상당수의 코드베이스를 공유
- SwiftUI와 함께 Cocoa, Core Foundation 등의 시스템 API를 제공
- macOS는 라이선스상 애플 기기 이외의 기기에서는 구동시킬 수 없다.
- 애플의 하드웨어가 아닌 다른 x86-64 컴퓨터에서 macOS를 구동시키는 것을 해킨토시라 부른다.

5. 운영체제의 명령어

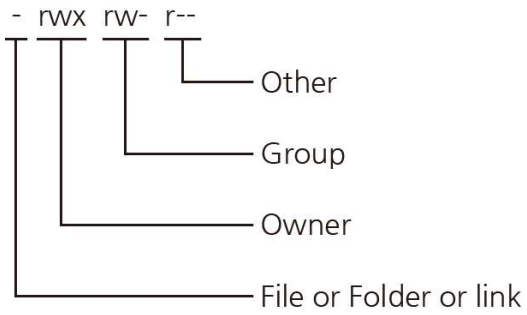
⑴ Linux, Unix 파일 접근 권한 관리

1) 파일 확인 방법

- ls -al 명령을 이용하여 파일의 상세정보 및 파일 접근 권한을 확인할 수 있다.

```
[root@db1 data]# ls -al
합 계  2959848
drwxrwxr-x 2 srtplay srtplay         51  1월   7  2021 .
drwxrwxr-x 4 srtplay srtplay         30  1월   7  2021 ..
-rw-r--r-- 1 srtplay srtplay 1438807183  5월  18  2020 dump_0515.0834.sql
-rw-rw-r-- 1 srtplay srtplay 1592073446  5월  28  2020 dump_0527.sql
```

2) 필드별 의미



1	2	3	4	5	6	7	8	9	10
-	R	W	-	R	-	-	R	-	-

① 1번 필드 : 타입

- 파일 : -
- 디렉터리 : d
- 다른 파일을 가리키는 링크 : l
- 두 개의 프로그램을 연결하는 파이프 파일 : p
- 블록 단위로 하드웨어와 반응하는 파일 : b
- 스트림 단위로 하드웨어와 반응하는 파일 : c

② 2~4번 필드 : 소유주(USER) 권한

③ 5~7번 필드 : 그룹(GROUP) 권한

④ 8~10번 필드 : 나머지(OTHER) 권한

3) 권한별 값

구분	값	설명
R	4	읽기 권한
W	2	쓰기 권한
X	1	실행 권한
-	0	권한 없음

4) 권한 변경

① chmod 0751 file명

② chmod 0775 file명

5) umask(접근 권한 마스크)

- 앞으로 만들어질 파일 권한에 대한 설정
- UNIX 에서는 파일은 666권한, 디렉터리는 777권한이 기본이다.
- umask로 지정한 8진수는 새로 만들어질 파일에서 제거될 권한을 명시

6) chown(소유주 변경)

- chown hungjik file명
  - 해당 파일의 소유주를 hungjik으로 변경

Section 2. 메모리 관리

1. 기억장치 관리 전략

⑴ 기억장치 관리 전략의 개념

- 보조기억장치의 프로그램이나 데이터를 주기억장치에 적재시키는 시기, 위치 등을 지정하여 한정된 주기억장치의 공간을 효율적으로 사용한다.

⑵ 기억장치 관리 전략

1) 반입(Fetch) 전략

- 보조기억장치에 보관 중인 프로그램이나 데이터를 언제 주기억장치로 적재할 것인지를 결정하는 전략

요구 반입 (Demand)	- 실행 중인 프로그램이 특정 프로그램이나 데이터 등의 참조를 요구할 때 적재하는 방법
예상 반입 (Anticipatory)	- 실행 중인 프로그램에 의해 참조될 프로그램이나 데이터를 미리 예상하여 적재하는 방법

2) 배치(Placement) 전략

- 새로 반입되는 프로그램이나 데이터를 주기억장치의 어디에 위치시킬 것인지를 결정하는 전략

최초 적합 (First Fit)	- 프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역 중에서 첫 번째 분할 영역에 배치
최적 적합 (Best Fit)	- 프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역 중에서 단편화가 가장 작은 분할 영역에 배치
최악 적합 (Worst Fit)	- 프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역 중에서 단편화가 가장 큰 분할 영역에 배치

3) 교체(Replacement) 전략

- 주기억장치의 모든 영역이 이미 사용 중인 상태에서 새로운 프로그램이나 데이터를 주기억장치에 배치하려고 할 때, 이미 사용되고 있는 영역 중에서 어느 영역을 교체하여 사용할 것인지를 결정하는 전략
- 종류 : FIFO, OPT, LRU, LFU, NUR, SCR 등

2. 단편화

⑴ 단편화의 개념

- 주기억장치에 프로그램을 할당하고 반납하는 과정에서 발생하는 사용되지 않는 작은 조각 공간

⑵ 단편화의 종류

1) 내부 단편화

- 주기억장치 공간이 프로그램보다 커서 프로그램의 사용 공간을 할당 후 사용되지 않고 남아있는 공간

2) 외부 단편화

- 주기억장치 공간보다 프로그램이 커서 프로그램이 할당될 수 없어 사용되지 않고 남아있는 공간

⑶ 단편화 해결 방법

1) 통합(Coalescing) 기법

- 인접해 있다면 두 개의 빈 분할 공간을 하나로 통합하여 효율성을 높이는 작업

2) 압축(Compaction) 기법

- 주기억장치 내 분산되어 있는 단편화 공간들을 통합하여 하나의 커다란 빈 공간을 만드는 작업

3) 재배치 기법(Relocation)

- 압축을 실행하여 이 과정에서 프로그램의 주소를 새롭게 지정해주는 기법

Section 3. 가상기억장치

1. 가상기억장치

⑴ 가상기억장치의 개념

- 보조기억장치(하드디스크)의 일부를 주기억장치처럼 사용하는 것

⑵ 블록 분할 방법

1) 페이징(Paging) 기법

- 가상기억장치를 모두 같은 크기의 블록으로 편성하여 운용하는 기법
- 외부 단편화는 발생하지 않으나 내부 단편화는 발생
- 주소 변환을 위해서 페이지 맵 테이블이 필요
- 페이지 크기별 비교

페이지 크기	기억장소 효율	단편화	입출력 시간	맵 테이블
클수록	감소	증가	감소	감소
작을수록	증가	감소	증가	증가

2) 세그먼테이션(Segmentation) 기법

- 가상 메모리를 서로 크기가 다른 논리적 단위인 세그먼트로 분할하고 메모리를 할당하는 기법
- 내부 단편화는 발생하지 않으나 외부 단편화가 발생
- 주소 변환을 위해서 세그먼트 맵 테이블이 필요

2. 가상기억장치 기타 관리사항

⑴ 페이지 부재

- 프로세스 실행 시 참조할 페이지가 주기억장치에 없는 현상

⑵ 지역성(Locality)

- 프로세스가 실행되는 동안 주기억장치를 참조할 때 일부 페이지만 집중적으로 참조하는 성질
- 지역성의 종류

시간 구역성 (Temporal Locality)	- 프로세스가 실행되면서 하나의 페이지를 일정 시간 동안 집중적으로 액세스하는 현상 - Loop(반복), Stack(스택), 부 프로그램(Sub Routine) 등
공간 구역성 (Spatial Locality)	- 프로세스 실행 시 일정 위치의 페이지를 집중적으로 액세스하는 현상 - 배열순회, 순차적 코드 실행 등

⑶ 워킹 셋(Working Set)

- 프로세스가 일정 시간 동안 자주 참조하는 페이지들의 집합

⑷ 스래싱(Thrashing)

- 프로세스의 처리 시간보다 페이지 교체에 소요되는 시간이 더 많아지는 현상
- 가상기억장치를 사용하는 시스템에서 하나의 프로세스 수행 과정 중 자주 페이지 부재가 발생함으로써 나타나는 현상으로, 전체 시스템의 성능이 저하됨

3. 페이지 교체 알고리즘

⑴ FIFO(First In First Out)

- 가장 먼저 메모리에 적재된 페이지를 먼저 교체하는 기법
- 프레임 개수를 늘리면 부재 발생이 감소해야 하나, 오히려 더 늘어나는 Belady's Anomaly 이상 현상 발생

⑵ OPT(Optimal replacement, 최적 교체)

- 앞으로 가장 사용되지 않을 페이지를 교체
- 페이지 부재 횟수가 가장 적게 발생하는 가장 효율적인 알고리즘이나 참조 상황을 예측하기 어려움

⑶ LRU(Least Recently Used)

- 최근에 가장 오랫동안 사용되지 않은 페이지를 교체

⑷ LFU(Least Frequently Used)

- 사용 빈도가 가장 적은 페이지를 교체

⑸ NUR(Not Used Recently)

- 최근의 사용여부를 확인하기 위해 각 페이지마다 두 개의 비트 사용
- 참조비트와 변형비트를 이용해서 페이지 교체

⑹ SCR(Second Chance Replacement)

- 가장 오랫동안 주기억장치에 있던 페이지 중 자주 사용되는 페이지의 교체를 방지하기 위한 것으로, FIFO 기법의 단점을 보완하는 기법

Section 4. 프로세스

1. 프로세스

⑴ 프로세스의 개념

- 컴퓨터에서 연속적으로 실행되고 있는 컴퓨터 프로그램

⑵ 스레드(Thread)

1) 스레드의 개념

- 프로세스 내에서 실행되는 흐름의 단위
- 프로그램은 하나 이상의 프로세스를 가지고 있고, 하나의 프로세스는 반드시 하나 이상의 스레드를 갖는다.

2) 스레드의 분류

① 사용자 수준의 스레드

- 사용자가 만든 라이브러리를 사용하여 스레드를 운용한다.
- 속도는 빠르지만 구현이 어렵다.

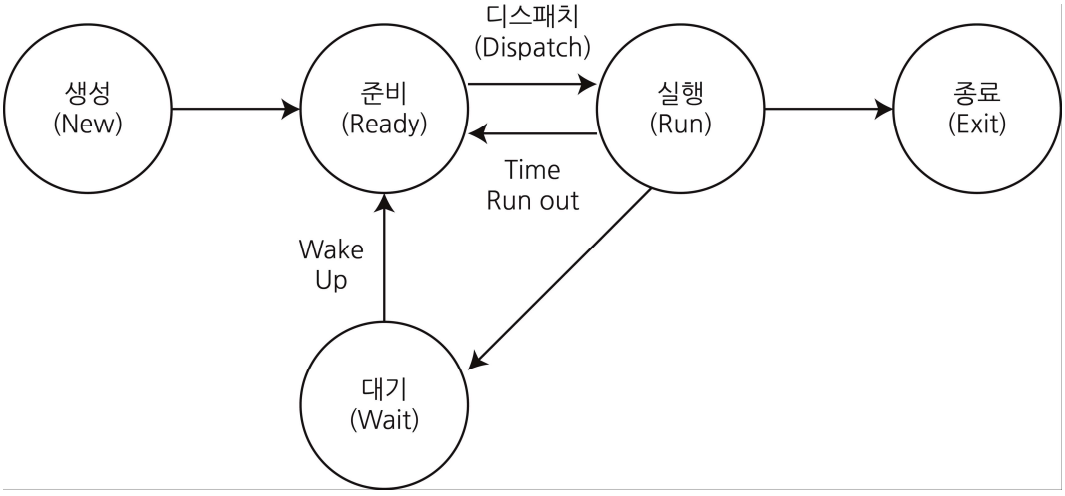
② 커널 수준의 스레드

- 운영체제의 커널에 의해 스레드를 운용한다.
- 구현이 쉽지만 속도가 느리다.

⑶ 메모리상의 프로세스 영역

프로세스 영역	설명
코드 영역	- 실행할 프로그램의 코드가 저장되는 영역 - 함수, 제어문, 상수 등이 지정
데이터 영역	- 전역변수와 정적변수가 할당되는 부분 - 프로그램이 종료되면 메모리에서 소멸
스택 영역	- 프로그램이 자동으로 사용하는 임시 메모리 영역 - 지역변수와 함수의 매개변수가 저장되는 영역으로 함수 호출이 완료되면 사라짐
힙 영역	- 프로그래머가 할당/해제하는 메모리 공간 - 동적 할당

⑷ 프로세스 상태 전이



⑸ PCB(Process Control Block, 프로세스 제어 블록)

- 운영체제가 프로세스에 대한 정보를 저장해 놓는 공간
- PCB에 저장되는 정보
  - ① 프로세스의 현재 상태
  - ② 포인터(부모 프로세스, 자식 프로세스 등)
  - ③ 프로세스 고유 식별자
  - ④ 스케줄링, 프로세스 우선순위
  - ⑤ CPU 레지스터 정보
  - ⑦ 주기억장치 관리 정보
  - ⑧ I/O 상태정보
  - ⑨ 계정정보(CPU 사용 시간, 실제 사용시간, 한정된 시간)

⑹ 문맥 교환(Context Switching)

- 하나의 프로세스가 CPU를 사용 중인 상태에서 다른 프로세스가 CPU를 사용하도록 하기 위해 이전의 프로세스의 상태를 PCB에 보관하고 또 다른 프로세스의 정보를 PCB에서 읽어 레지스터에 적재하는 과정
- 문맥 교환은 멀티태스킹(=멀티프로세싱)이 가능하도록 해준다.

2. 프로세스 스케줄링

⑴ 스케줄링(Scheduling)의 개념

- 메모리에 올라온 프로세스들 중 어떤 프로세스를 먼저 처리할지 순서를 정하는 것

⑵ 스케줄링의 목적

목적	설명
공평성	모든 프로세스가 자원을 공평하게 배정받아야 하며, 특정 프로세스가 배제되어서는 안 된다.
효율성	시스템 자원을 최대한 활용하여 스케줄링 해야 한다.
안정성	중요한 프로세스가 먼저 처리되도록 한다.
반응 시간 보장	적절한 시간 안에 프로세스의 요구에 반응해야 한다.
무한 연기 방지	특정 프로세스의 작업이 무한하게 연기되어서는 안 된다.

⑶ 스케줄링 기법

1) 선점형 스케줄링(Preemptive)

- 다른 프로세스가 실행 중이더라도 운영체제가 CPU를 강제로 뺏을 수 있는 방식
- 종류 : Round Robin, SRT, 다단계 큐(MLQ, Multi-Level Queue), 다단계 피드백 큐(MLFQ, Multi-Level Feedback Queue) 등

2) 비선점형 스케줄링(Non-Preemptive)

- 프로세스가 CPU를 점유하고 있다면 이를 빼앗을 수 없는 방식
- 순서대로 처리되는 공정성 보장
- 종류 : FCFS, SJF, HRN, 우선순위, 기한부 등

3) 기아현상과 에이징 기법

종류	설명
기아현상 (Starvation)	- 시스템에 부하가 많아서 우선순위가 낮은 프로세스가 무한정 기다리는 현상 - SJF, 우선순위, SRT, MLQ
에이징 기법 (Aging)	- 기아현상을 해결하기 위한 기법 - 오랫동안 기다린 프로세스에게 우선순위를 높여주는 기법 - HRN, MLFQ

3. 스케줄링 알고리즘

⑴ 선점형 기법

1) Round Robin
- 시간단위(Time Quantum/Slice)를 정해서 프로세스를 순서대로 CPU를 할당하는 방식

2) SRT(Shortest Remaining Time)
- 비선점 스케줄링인 SJF 기법을 선점 형태로 변경한 기법
- CPU 점유 시간이 가장 짧은 프로세스에 CPU를 먼저 할당하는 방식

3) 다단계 큐(MLQ, Multi-Level Queue)
- 프로세스를 특정 그룹으로 분류할 수 있을 경우 그룹에 따라 각기 다른 준비 상태 큐를 사용하는 기법

4) 다단계 피드백 큐(MLFQ, Multi-Level Feedback Queue)
- 프로세스 생성 시 가장 높은 우선순위 준비 큐에 등록되며 등록된 프로세스는 FCFS 순서로 CPU를 할당받아 실행되고, 할당된 시간이 끝나면 다음 단계의 준비큐로 이동
- 단계가 내려갈수록 시간 할당량이 증가하고, 가장 하위큐는 Round Robin 방식으로 운영

⑵ 비 선점형 기법

1) FCFS(First Come First Serve)
- 먼저 도착한 프로세스를 먼저 처리하는 스케줄링 알고리즘

2) SJF(Shortest Job First)
- 실행시간이 가장 짧은 프로세스에게 CPU를 할당하는 방식

3) HRN(Highest Response ratio Next)
- SJF 기법에서 비교적 실행시간이 긴 프로세스가 가질 수 있는 불리함을 보완한 스케줄링 방식
- 우선순위 = (대기시간 + 실행시간) / 실행시간

4) 우선순위(Priority)
- 프로세스마다 우선순위를 부여하여 높은 우선순위를 가진 프로세스에게 먼저 자원을 할당

5) 기한부(Deadline)
- 프로세스에게 일정한 시간을 주어 그 시간 안에 완료하도록 하는 기법
- 주어진 시간 내에 완료되지 못할 경우, 해당 프로세스는 제거되거나 처음부터 다시 실행되어야 하므로 부담이 매우 큰 기법

Section 5. 병행 프로세스와 교착상태

1. 병행 프로세스

⑴ 병행 프로세스의 개념

- 두 개 이상의 프로세스들이 동시에 존재하며 실행상태에 있는 것

⑵ 문제점과 해결책

1) 문제점

- 동시에 2개 이상의 프로세스를 병행 처리하면 한정된 자원(CPU, 메모리, 디스크, I/O 장치 등)에 대한 사용 순서 등 여러 가지 문제가 발생

2) 문제 해결책

- 임계구역
- 상호배제 기법
- 동기화 기법

2. 병행 프로세스 문제 해결책

⑴ 임계구역(Critical Section)

- 공유 자원에 대해서 한 순간에는 반드시 하나의 프로세스만 사용되도록 지정한 영역

⑵ 상호 배제(Mutual Exclusion)

- 하나의 프로세스가 공유 메모리 혹은 공유 파일을 사용하고 있을 때 다른 프로세스들이 사용하지 못하도록 배제시키는 제어 기법
- 상호 배제 기법
  ① 데커의 알고리즘(Dekker's Algorithm)
  ② 피터슨의 알고리즘(Peterson's Algorithm)
  ③ 다익스트라 알고리즘(Dijkstra Algorithm)
  ④ 램포트의 베어커리 알고리즘(Lamport's Bakery Algorithm)

⑶ 동기화 기법

- 스레드들에게 하나의 자원에 대한 처리 권한을 주거나 순서를 조정해주는 기법

1) 세마포어(Semaphore)

- 각 프로세스에 제어 신호를 전달하여 순서대로 작업을 수행하도록 하는 기법

2) 모니터(Monitor)

- 프로그래밍 언어 수준에서 동시성을 제어하여 타이밍 오류를 해결한 상호 배제 기법

3. 교착상태(Dead Lock)

(1) 교착상태의 개념

- 상호 배제에 의해 나타나는 문제점으로, 둘 이상의 프로세스들이 자원을 점유한 상태에서 서로 다른 프로세스가 점유하고 있는 자원을 요구하며 무한정 기다리는 현상

(2) 교착상태 발생 조건

발생 조건	설명
상호배제 (Mutual Exclusion)	한 번에 한 개의 프로세스만이 공유 자원을 사용할 수 있어야 함
점유와 대기 (Hold & Wait)	자원을 점유하고 있으면서 다른 프로세스에 할당되어 있는 자원을 추가로 요구하며 대기
비선점 (Nonpreemption)	프로세스에 할당된 자원은 사용이 끝날 때까지 강제로 빼앗을 수 없음
환형대기 (Circular Wait)	각 프로세스가 순차적으로 다음 프로세스가 요구하고 있는 자원을 가지고 있는 상태

(3) 교착상태 해결 방법

해결 방법	설명
예방 기법 (Prevention)	- 교착 상태가 발생되지 않도록 사전에 시스템을 제어하는 방법
회피 기법 (Avoidance)	- 교착 상태가 발생하려고 할 때 교착상태 가능성을 피해가는 방법 - 주로 은행원 알고리즘(Banker's Algorithm)이 사용
발견 기법 (Detection)	- 시스템에 교착 상태가 발생했는지 점검하여 교착 상태에 있는 프로세스와 자원을 발견하는 것
회복 기법 (Recovery)	- 교착상태의 프로세스에 할당된 자원을 선점하여 프로세스나 자원을 회복하는 것 - 교착 상태를 일으킨 프로세스를 종료시킴으로 선점을 해제한다.

Section 6. 디스크 스케줄링(Disk Scheduling)

1. 디스크 스케줄링

⑴ 디스크 스케줄링 개념

- 사용할 데이터가 디스크상의 여러 곳에 저장되어 있을 경우, 데이터를 액세스하기 위해 디스크 헤드를 움직이는 경로를 결정하는 기법

⑵ 디스크 스케줄링 종류

1) FCFS 스케줄링(First Come First Service)
- 요청이 들어온 순서대로 처리

2) SSTF(Shortest Seek Time First)
- 현재 헤드에서 가장 가까운 트랙의 요청을 먼저 처리한다.

3) SCAN
- 헤드의 진행방향에 있는 요청을 처리하고, 다시 반대방향으로 틀어 반대방향에 있는 요청들을 처리한다.
- 진행되는 과정에서 요청이 들어오면 해당 요청도 처리한다.

4) C-SCAN
- 항상 한쪽 방향에서 반대방향으로 진행하며 트랙의 요청을 처리한다.
- 바깥쪽에서 안쪽으로 진행하며 요청을 처리한다.
- 진행되는 과정에서 요청이 들어오면 해당 요청은 처리하지 않는다.

5) LOOK
- SCAN 기법을 기초로 사용하며, 진행 방향의 마지막 요청을 처리한 후 반대방향으로 처리하는 기법

6) C-LOOK
- C-SCAN 기법을 기초로 사용하며, 바깥쪽에서 안쪽 방향의 모든 요청을 처리한 후 가장 바깥쪽으로 이동한 뒤에 다시 안쪽 방향으로 서비스하는 기법

7) N-STEP SCAN
- SCAN 기법을 기초로 두고 있으며, 시작하기 전 대기하고 있는 요청들을 우선적으로 처리하고, 처리하는 과정 중에 요청이 들어오는 것들은 이후에 모아서 반대방향으로 진행할 때 서비스한다.

8) 에션바흐(Eschenbach)기법
- 부하가 매우 큰 항공 예약 시스템을 위해 개발
- 탐색 시간과 회전 지연 시간을 최적화하기 위한 최초의 기법

Section 7. 환경변수와 로그 파일

1. 환경변수

⑴ 환경변수의 개념

- 프로세스가 컴퓨터에서 동작하는 방식에 영향을 미치는 동적인 값들의 모임

⑵ UNIX/Linux 환경변수

- env, set, printenv 명령어들을 사용하여 환경변수와 그에 따른 모든 값을 볼 수 있다.
- export 명령을 이용하여 사용자 환경변수를 전역변수로 설정할 수 있다.

⑶ Windows 환경변수

- 제어판 > 시스템 및 보안 > 시스템 > 고급 시스템 설정 > 환경변수 존재
- 커맨드 창에서 set 명령으로 확인

2. 로그 파일

⑴ 로그의 개념

- 시스템의 모든 기록을 담고 있는 데이터

⑵ 리눅스 로그 종류

종류	설명
messages	시스템 로그 파일
secure	보안인증에 관한 메시지 로그파일
maillog	메일 로그 파일
xferlog	ftp 로그파일
dmesg	부팅 시의 시스템 로그
wtmp	시스템에 로그인 기록이 저장되는 파일(전체 로그인 기록)
utmp	시스템에 로그인 기록이 저장되는 파일(현재 로그인 사용자에 대한 기록)
btmp	로그인 실패 정보 기록
lastlog	각 계정들의 가장 최근 로그인 기록

Section 8. 스토리지

1. 스토리지(Storage)

⑴ 스토리지 개념

- 컴퓨터에 데이터를 저장하는 저장소의 역할을 수행하는 부품

⑵ 스토리지 종류

종류	설명
DAS (Direct Attachted Storage)	- PC나 서버에 직접 꽂아서 사용하는 스토리지
NAS (Networt Attached Storage)	- 서버와 저장장치가 이더넷 등을 이용해 네트워크에 연결된 방식
SAN (Storage Area Network)	- 서버와 저장장치를 파이버채널 스위치(광채널)로 연결한 고속데이터 네트워크 - 전용 파이버채널 스위치(광채널)를 둠으로써 빠른 속도의 연결을 유지한다.

⑶ RAID(Redundant Array of Inexpensive Disks)

1) RAID 개념

- 복수의 하드디스크를 하나의 드라이브와 같이 인식하고 표기한다.

2) RAID 구성

① 스트라이핑(Striping)

- 논리적으로 연속된 데이터들이 물리적으로 여러 개의 디스크에 라운드로빈 방식으로 저장되는 형태

② 미러링(Mirroring)

- 데이터를 그대로 복제하는 것으로 신뢰성 및 가용성 확보를 위해 사용됨

3) RAID 형태

형태	설명
RAID-0	- 빠른 데이터 입출력을 위해 스트라이핑을 사용하는 방식으로, 디스크의 모든 용량을 사용한다. - 하나의 디스크가 잘못되면 데이터를 잃어버릴 수 있다.
RAID-1	- 두 개 이상의 디스크를 미러링을 통해 하나의 디스크처럼 사용한다. - 완전히 동일하게 데이터를 복제하기 때문에 가용량이 절반이다. - 하나의 디스크에서 에러가 발생하면 미러링된 디스크를 통해 복구가 가능하다.
RAID-2	- 오류 정정을 위한 해밍코드를 사용하는 방식
RAID-3	- 하나의 디스크는 패리티(Parity) 정보를 위해 사용하고 나머지 디스크에 데이터를 균등하게 분산 저장 - 하나의 디스크에서 에러가 발생하면 패리티 디스크를 통해 복구할 수 있다.
RAID-4	- RAID-3과 같이 패리티 정보를 독립된 디스크에 저장한다. - 블록(Block)단위로 분산 저장하는 차이가 있다.

RAID-5	- 3개 이상의 디스크를 붙여서 하나의 디스크처럼 사용하고 각각의 디스크에 패리티 정보를 가지고 있는 방식 - 패리티 디스크를 별도로 사용하지 않음으로 병목현상이 발생하지 않는다.
RAID-6	- 하나의 패리티를 두 개의 디스크에 분산 저장하는 방식 - 패리티를 이중으로 저장하기 때문에 두 개의 디스크에 에러가 발생해도 정상적으로 복구할 수 있다.