Ch1) Introduction

1. \*\*\* What is an operating system? And describe 5 main management components of an Operating System

* 운영체제는 사용자가 컴퓨터 하드웨어를 효율적으로 관리해서 사용자가 컴퓨터 시스템을 편리하게 사용할 수 있도록 해 준다.
* 5가지 main Management 구성요소에는 Process Management, Main memory management, I/O system management, secondary storage management, file management 가 있다.

1. \*\*\* System software layers

* Computer system architecture -> operating system -> software development environment-> (middleware, 중간소프트웨어) -> user applications
* Operating system = kernel + system program(운영체제와 함께 깔림 ex)윈도우 탐색기)

1. I/O device controller는 어떤 역할을 하는가?

* I/O기기들은 연산 속도가 매우 느리기 때문에 CPU와 속도를 맞추기 위해 controller가 꼭 필요하다.

1. \*\*\* Memory hierarchy(메모리 계층 구조)  
   magnetic tape->optical disk->hard-disk drives->nonvolatile memory->main memory->cache->registers  
   위로 갈수록 cpu와 가까히 있어서 속도가 빠르고 아래로 갈수록 거쳐야 하는 계층이 많기 때문에 느려진다. 경제성 때문에 생긴 구조.
2. CPU
3. Registers (cpu의 저장소)

- Program counter(PC), Instruction Register(IR), \*Program Status Word(PSW)

(2) \*Pipelining (특수한 작업을 병렬로 처리하도록 한 하드웨어 기법)

- Fetch, Decode, Execute, Write Back

1. \* Caching (똑같은 데이터를 100번 부른다면?)

* Use of high-speed memory to hold recently-accessed data

1. Describe the operation of interrupt in details. 아래 8, 9, 10 답변
2. I/O control  
   Interrupt: cpu의 즉각적인 처리를 필요로 하는 이벤트를 알리기 위해 하드웨어나 소프트웨어로 부터 요청을 받는 것  
   - Polling: 정기적으로 cpu의 상태를 확인하는 것(오버헤드가 큼)  
   - hardware interrupt: I/O 장치로부터 작업이 다 끝났다는 것을 보고받는 방식(효율적, 오버헤드가 적음, 주로 사용)
3. I/O control 순서  
   디바이스 인터럽트 -> cpu에서 현재 수행중인 것 중지, 보존-> 인터럽트 루틴 수행 -> 상태를 복구, 중단된 프로세스 재개
4. \*\* Interrupts and Exceptions 의 차이는?

(1) Interrupts -> I/O장치가 cpu에게 이벤트가 발생했다는 사실을 전달함

a. Polling: 지속적으로 cpu에 물어봄(장치 작업 다 끝남?)

b. Hardware interrupt: 장치에서 알려줌(주로 이 방식 사용)

1. Exceptions -> CPU가 자기 자신에게 interrupts를 건다.
2. trap(expected)-> 의도적으로 발생시킴(어플리케이션이 운영체제에 서비스를 요청, ex) system call )
3. Fault(unexpected)->예상치 못한 상황 (ex)참조할 수 없는 메모리)
4. Hardware Protection

* I/O Protection
* Memory Protection: 자신에 할당되지 않은 메모리 참조 방지
* CPU Protection: 하나의 프로그램이 cpu를 독차지하지 못하도록 방지 (Timer interrupt)

1. Dual mode operation -> User mode, Kernel mode

* CPU를 보호하기 위해 듀얼모드 사용

1. \*\*Privileged instructions can be issued only in monitor mode(=kernel mode). ex) I/O instruction은 privileged instruction임
2. Computer system에는 mainframe systems, simple batch systems, multiprogrammed batch system 등등이 있다. 이중 cpu 스케줄링이라는 개념이 등장하기 시작한건 언제인가?

* multiprogrammed batch system

1. Concurrent system과 Parallel system의 차이는?

* Concurrent: CPU가 1개, 여러 개의 프로세스를 동시에 수행하는 것처럼 처리(사실은 번갈아서 수행)
* Parallel: CPU가 여러 개, 각각 여러 개의 프로세스를 맡아서 동시에 처리.

1. Parallel system의 symmetric multiprocessing(SMP)와 asymmetric multiprocessing 의 차이는?

* Symmetric multiprocessing(SMP): 똑 같은 CPU 여러 개 사용
* Asymmetric multiprocessing: 좋은 CPU1개, 평범한 CPU 여러 개 사용

ch2) System structures

1. What are the meanings of “user mode” And “kernel mode” in OS? and describe how to access to kernel mode from user mode  
   - User mode는 사용자가 일반적으로 컴퓨터를 사용할 때의 모드이고 Kernel mode는 운영체제가 컴퓨터 제어를 하는 모드이다. User mode에서 kernel mode로 접근하기 위해서는 system call이 필요하다. Read()와 같은 system call 함수를 통해 kernel mode로 진입 후 운영체제에서 이를 처리해준다. 위와 같은 방식으로 user mode와 kernel mode는 system call interface를 통해 정보를 처리한다.
2. System call  
   - 운영체제에 접근하기 위한 함수  
   - user mode, kernel mode  
   - system call interface를 통해 커널모드로 진입, 시스템 제어
3. Function call과 System call 의 차이는?

* 사용자가 만든 함수를 호출, 운영체제가 만들어 놓은 함수를 호출

1. Program과 Process의 차이는?

* Program 은 그냥 실행’파일’임(XX.exe)
* Process는 program의 인스턴스

1. \*\*Monolithic Kernel 의 장단점은? (Unix, Linux)

* 장점: 하나로 통합되어 있어서 각 component간의 통신이 효율적
* 단점: 특정한 드라이버를 추가하거나 삭제하려면 커널을 재빌드 해야함, 기능을 추가할수록 무거워져서 활용도가 떨어짐.(embedded system에서 사용하기 힘듬)?? , 복잡하게 얽혀있기 때문에 관리하기 힘듬(software engineering issue가 있다)

1. \*\*MicroKernel 의 장단점은?(Mac OS)

* 중요한 서비스들은 커널의 밑단에 넣고 덜 중요한 서비스들은 user mode에서 각각 server형태로 관리한다. Monolithic kernel의 문제점을 해결하기 위해 탄생.
* 장점: 각 서버를 추가하는 방식이기 때문에 기능을 추가하기 쉽다. 시스템이 견고하고, Real-time성이 높다.
* 단점: 각 서버의 communication에 오버헤드가 있다는 것이 단점.

1. I/O system management -> 개념적으로 중요

* A buffer-caching system, a general device-driver interface, i/o abstraction

Ch3) Process Concept

* An instance of a program in execution
* Program(실행파일)이 메모리에 적재될 때 프로그램은 프로세스가 된다.

1. Process address space를 구성하는 4가지 영역?

* Stack(dynamic allocated memory)- ex)지역변수, 매개변수(함수 호출)
* Head(dynamic allocated memory)- ex)동적 메모리 할당
* Static data(data segment)- ex)전역변수(변화하지 않는 데이터, 프로그램이 종료되어야 없어짐
* Code data(text segment)-text

1. Process state의 5가지 상태를 나열하세요.

* New: 프로세스(process)가 생성됨
* Running: 명령어(instruction)가 수행됨
* Waiting: 이벤트(I/O interrupt)가 발생하여 프로세스가 기다리고 있는 상태
* Ready: Running상태가 되기전에 준비가 된 상태
* Terminated: 프로세스가 종료됨

1. Scheduler dispatch가 무엇?

* CPU에게 자원할당을 하여 Ready 상태에 있는 process를 running 상태로 만드는 것.

1. Process Control Block(PCB)이 가지고 있는 정보는?

* 프로세스의 정보를 블록 단위로 저장
* +(<https://kldp.org/node/26022> <- pointer와 program counter의 차이에 대해 설명)

1. Context switch는 오버헤드가 발생한다는 단점을 가지고 있다. 그럼에도 사용하는 이유는?

* Overhead의 단점 << Timesharing의 장점

1. Process Scheduling Queue에는 Job queue, Ready queue, Device queue 등이 있다.
2. Long-term scheduler(job scheduler)와 \*\*short-term scheduler(CPU scheduler)의 차이는?
3. long-term scheduler는 최근에 사라졌다. 그 이유는?

* 가상메모리 등장(프로세스를 모두 메모리에 올려놓을 필요가 없음, 순간 프로그램에 필요한 프로세스만 메모리에 올려놓는다.)

1. \*process creation – fork() 함수를 사용하면 process는 몇 개가 될까?

* 2개 (원래 있던 부모 process, 자식 process)
* Fork()함수는 부모 프로세스를 복사하여 자식 프로세스를 생성하는 개념이다.

1. fork()함수의 실행 과정
2. Creates and initializes a new PCB
3. Creates and initializes a new address space
4. \*\*Initializes the address space with a copy of the entire contents of the address(부모 프로세스의 모든 정보를 새로 할당된 공간에 카피)
5. \*\*initializes the kernel resources to point to the resources used by parent(부모프로세스의 커널 리소스를 자식프로세스가 사용할 수 있게 해줌)
6. Places the PCB on the ready queue
7. Returns the child’s PID to the parent, and zero to the child
8. Independent process and Cooperating Processes
9. Independent process – 독립프로세스
10. Cooperating process – 협력프로세스
11. Inter-Process Communication(IPC) – 실제 프로세스가 통신하는 방식
12. Message passing

* 장점: Process A와 Process B가 수행될 때 동기화를 맞춰주는 작업이 필요한데 이것을 커널이 알아서 해줌.
* 단점: A에서 B로 데이터를 옮길 때 데이터의 크기가 크다면 운영체제를 거쳐서 가야하기 때문에 오버헤드가 발생

1. Shared memory

* 장점: Shared memory 생성만 운영체제가 하고 두 프로세스가 Shared memory를 통해 통신하기 때문에 오버헤드 발생X
* 단점: 두 Process간의 데이터 동기화 작업을 kernel에서 해주는 것이 아닌 직접 프로그래밍 하면서 동기화를 해줘야함( 번거로움, 동기화 작업을 신경써줘야함)

1. Process A가 ProcessB의 영역에 직접 접근해서 통신하는 방식은 불가능하다. 왜?

* Memory protection
* Producer – Consumer Problem
* Producer는 데이터를 생성, 넣기만 하고 consumer는 버퍼에 저장된 데이터를 꺼내 쓰기만함

1. Bounded-Buffer: Shared-Memory solution

* Buffer는 circular queue로 구현됨, 이 buffer가 shared memory가 된다.
* 버퍼의 사이즈를 픽스하는 것
* 이렇게 shared memory는 queue가 꽉 찾는지를 확인하는 코드를 직접 개발자가 넣어줘야 하지만, 운영체제의 queue를 사용하면 알아서 해준다.

1. Direct communication and Indirect communication

* Direct communication: 메시지를 수신할 process지정
* Indirect communication: 수신할 특정 process지정X(mail box에 넣어둠, 찾아가~)

1. Synchronous and Asynchronous

* Synchronous(Blocking): 데이터가 도착하기 전에 어떤 다른 프로세스가 접근하는 것을 막음
* Asynchronous(non-blocking)
* 기본적으로 Blocking 방식을 사용함

1. Buffering
2. Zero capacity: 0 messages(더 이상 버퍼에 데이터를 저장하지 않음. 누군가가 데이터를 가져가려고 하면 버퍼에 있는 것을 바로 writing해버리고, 가져가려고 하지 않으면 바로 삭제한다.
3. Bounded capacity: finite length of n messages
4. Unbounded capacity: infinite~
5. What is Inter-Process Communication (IPC)? And describe detail of the method of IPC.

* 두개의 프로세스가 자원을 공유하려면 서로 데이터를 주고 받을 수 있는 방식이 필요하다. 직접 서로의 메모리에 접근하는 것은 Memory protection때문에 불가능하다. 그래서 Inter-process communication이 등장했는데, IPC 방식에는 Message passing과 Shared memory 두가지 방법이 있다. Message passing은 두 개의 process가 서로 정보를 전달할 때 kernel-mode를 거쳐서 전달하는 방식이고 shared-memory는 같은 상황일 때 user-mode에서 서로 정보를 전달할 공유 메모리를 만들어서 보다 직접적으로 통신하는 방식이다. Message passing은 운영체제 kernel을 거쳐서 가기 때문에 오버헤드가 크고 느리다는 단점이 있지만 데이터를 전달만 하면 운영체제가 알아서 해준다는 장점이 있다. shared-memory는 반대로 kernel을 거치지 않기 떄문에 오버헤드가 적고 속도가 비교적 빠르다는 장점이 있지만 공유 자원인 shared-memory 공간을 직접 개발자가 구현해야 한다는 번거로움이 있다.