Ch1) Introduction

1. What is an operating system? (정의)

* A program that acts as an intermediary between a user of a computer and the computer hardware
* Use the computer hardware in an efficient manner

1. Computer system components? (컴퓨터 시스템의 4가지 구성요소)

* User, application programs, operating system, computer hardware
* Operating system = kernel + system program(운영체제와 함께 깔림 ex)윈도우 탐색기)

1. I/O device 에 컨트롤러가 필요한 이유는?

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

1. CPU
2. 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. \*\* Interrupts and Exceptions 의 차이는?

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

* Polling: 지속적으로 시스템에 물어봄(장치 작업 다 끝남?)
* Hardware interrupt: 장치에서 알려줌(주로 이 방식 사용)

1. Exceptions -> CPU가 자기 자신에게 interrupts를 건다.

* Trap(expected)-> 의도적으로 발생시킴(어플리케이션이 운영체제에 서비스를 요청, ex) system call )
* Fault(unexpected)->예상치 못한 상황 (ex)참조할 수 없는 메모리)

1. 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
* \*\*Privileged instructions can be issued only in monitor mode(=kernel mode). ex) I/O instruction은 privileged instruction임

1. 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 수행하는것처럼 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. Function call과 System call 의 차이는?

* 사용자가 만든 함수를 호출, 운영체제가 만든 함수를 호출

1. Program과 Process의 차이는?

* Program 은 그냥 실행파일임(XX.exe)
* Process는 프로그램의 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
* 실행파일이 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(text segment)-

1. Process state의 5가지 상태를 나열하세요.

* New, running, waiting, ready, terminated

1. Schedule dispatch가 무엇?

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

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

* Process state, Process ID, pointer, program counter, cpu registers ..
* + (<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
* Process Scheduling Queue에는 Job queue, Ready queue, Device queue 등이 있다.

1. Process가 running 상태에 놓이기 직전에 어떤 큐에 들어가 있을까?

* Ready Queue

1. Device Queue에 있는 process의 작업이 끝나면 어떤 큐에 추가될까?

* Ready Queue

1. Long-term scheduler(job scheduler)와 \*\*short-term scheduler(CPU scheduler)의 차이는?
2. 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)

1. \*int 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. Int exec()함수의 실행 과정
9. Stops the current process
10. Loads the program”prog” into the process’ address space
11. Initializes hardware context and args for the new program
12. Places the PCB on the ready queue(\*note: exec() does not create a new process) -> 실행은 CPU Scheduling에 따라
13. What does it mean for exec() to return?->ready queue에 잘 올라가져 있나 없나 확인
14. Independent process and Cooperating Processes
15. Independent process – 독립프로세스
16. Cooperating process – 협력프로세스
17. Inter-Process Communication(IPC) – 실제 프로세스가 통신하는 방식
18. 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

1. 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~