운영체제 2020-2학기 중간고사 정리

고려대 컴퓨터정보통신대학원 빅데이터융합학과 조송현

chapter1.

OS는 resource allocator 이면서 control program 이다

Kernel : one program running at all times on the computer

bootstrap program: : loaded at power-up or reboot / typically stored in ROM or EEPROM, i.e.Firmware / initialize all aspects of system / OS 커널을 load하고, start execution

인터럽트 되면 transfer control to Interrupt service routine / interrupted instruction의 주소는 보존 / 인터럽트 작업 serviced 되면 saved return address loaded to program counter -> resume computation / OS는 interrupt-driven

trap or exception : interrupt caused by errror or user request

Caching: copying information into Faster storage system

Storage-Device Hierarchy: registers -> cache -> main memory -> nonvolatile memory -> hard-disk drives -> optical disk -> magnetic tapes

일반적으로 컴퓨터는 CPU, multiple device controllers로 구성되며 common bus 통해 연결 / OS는 각 device controllers 위한 device driver 갖고 있다 / I/O 시작되고나면 I/O completion되어야 control return to user program / Wait instruction은 다음 인터럽트 때까지 CPU를 idle 만든다 / 한번에 하나의 I/O Processing /

Multiprocessor system의 장점: throughput 증가 / economy in scale / reliability 증가(하나의 프로세스 실패가 시스템을 멈추지않는다)

Clustered Systems: 멀티프로세서시스템과 같이 multiple systems work together / Storage-area network(SAN)통해 스토리지 공유 / some are for high-performance computing(HPC) / some have distributed lock manager (DLM)

Progam is Passive entity, Process is Active entity

Single threaded process have one program conter (specify location to execute next instructions)

Memory management: keep track / decide which process to move into/out of memory / allocating&deallocating memory space

cache coherency: all CPUs have the most Recent value in their cache

distributed system : collection of physically separate&heterogeneous systems with access to various resources

Peer to Peer computing : 모든 노드는 peers로 본다 / clients와 servers를 구분하지 X

chapter 2.

OS services : user inetrface / Program execution / I.O operations / File-system manupulation / communications (via shared memory, message passing) / Resource allocation / Logging / Protection and security

System Calls : provide interface to servers / direct system calls 보다는 API가 선호됨 / API specify set of functions , include parameters, return values / Win32 API for Windows, POSIX API for UNIX,Linux,macOS, JAVA API / 왜 API를 사용하냐: program portability, easier

RTE(Run-Time Environment) provides a system-call interface (that serves as the link to system calls made available by the OS)

3가지 매개변수전달 방법(to pass parameters to the OS) : registers에서 전달 (가장 간단) / 블럭 또는 테이블 에 변수 저장하여 전달 / stack에 위치시켜 전달

Type of System Calls : Process Control, File Management, Device Management, Information Maintenance, Device management, Information maintenance, Communications, Protections

Arduino : single-tasking system / Simple hardware platform consisting of micocontroller

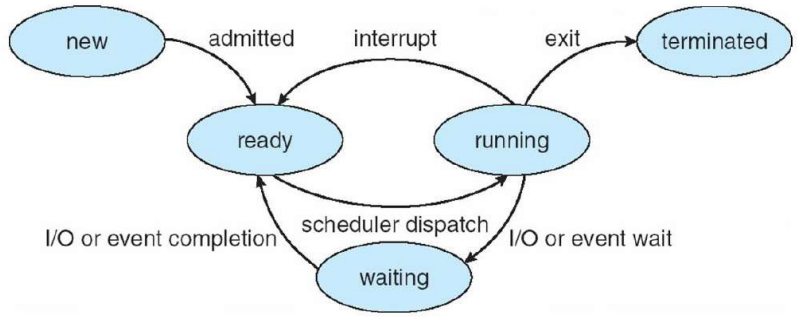
FreeBSD: Multitasking system / to start new process, shell executes fork() system

compile(into object file) -> Linker(combine relocatable object files into single binary executable file) -> Loader(to load binary executable file into memory)

Monolithic Structure: UNIX (system programs와 kernel로 구성) / Linux 또한 / difficult / 오버헤드 거의 없음 / kernel내 커뮤니케이션 빠름 / speed and efficiency

Microkernel: 클라이언트 프로그램 간 커뮤니케이션 제공, 유저스페이스에서 running하는 다양한 서비스 제공 / extending OS 쉽게 함 / 커널 수정 필요X / easier to port from / overhead증가(단점)

chapter 3.



PCB: 각 프로세스와 관련되는 many pieces of informations를 포함한다 / task control block이라고도 한다

process state: new, ready, running, waiting, halted, etc. / Program counter: 다음 명령어 수행될 주소 / CPU registers / CPU scheduling information(priority, pointer 등) / Memory management information / accounting information / I/O status information /

Process Scheduling : CPU utilization 을 극대화 위한 / time sharing의 목적은 CPU core를 switch 하여 users가 각 프로그램 수행중일 때 interact하는 것

Long-term scheduler : job scheduler / select process from this pool & load into memory for execution

Short-term scheduler: CPU scheduler / select from among processes (that are ready to execute and allocate CPU to one of them)

I/O bound process : computation보다 I/O에 spend time / many short CPU bursts / 모든 프로세스가 I/O바운드면 레디큐가 empty / short-term scheduler가 할 일 없어짐

CPU-bound process: computation에 spend more time / 모두 cpu바운드면 i/o 웨이팅큐가 empty, device unused, system unbalanced

Swapping – reintroduced into memory/ execution continue where it left off / necessary when memory overcommitted and must be freed up

Context switch : CPU를 다른 프로세스로 스위치하는 건 현 프로세스를 save, 그리고 다른 프로세스를 restore (storing process이다. restored and resume execution later) / 스위치 타임은 순수한

오버헤드 / 스위치타임 정도는 하드웨어 지원에 좌우됨

Process Creation : creating process는 parent process라 불리고 새로운 프로세스는 그 프로세스의 children / 대부분의 OS는 unique process identifier에 따라 프로세스를 identify한다 / UNIX를 예로 들면 fork()시스템콜로 새 프로세스 생성, exec()통해 replace process memory space with new program, exec()시스템콜은 메모리에 binary file을 로드함, variable pid 값은 child 프로세스에서 zero , parent는 0보다 큰 정수, parent는 child process 완료할때까지 wait() system call로 기다림, child process가 exit()으로 끝나면 parent process resume

Process Termination: final statement를 끝내고 OS에게 exit()을 이용하여 delete it하도록 요청하면 프로세스 종료됨. parent는 children 중 하나를 abort()로써 종료할 수 있음, wait(): pid를 반환하면서 child process가 끝나기를 기다리는 것.

Interprocess Communication(IPC): message passing (프로세스 커뮤니케이션 메커니즘, 프로세스 동작을 same address space를 셰어함으로써 동기화시킴). communication link가 있어야 하고 send/receive 통해 메시지를 교환해야 함 /

Indirect Communications : mailbox 또는 port통해 메시지 주고받고 /

Buffering : communication이 direct or indirect 관계없이 messages는 temporary queue에 존재하는 커뮤니케이션에 의해 교환됨.이러한 큐는 zero capacity, bounded(finite) ~, Unbounded(infinite) ~에 의해 실행됨.

Advanced Local Procedure Call(ALPC): 메시지 전달 facility in Windows

chapter 4.

thread : a light-weight process , CPU utilization의 기본단위 / 동일 프로세스 코드,데이터,OS자원 영역에 속하는 스레드와 share한다 / application은 수많은 스레드 제어와 각각의 프로세스로서 실행된다

Multicore Programming : 복수의 컴퓨팅 코어와 개선된 컨커런시 효율적 사용 위한 메커니즘ㅇㄹ 제공한다. 컨커런시는 스레드 실행이 interleaved되는 것을 의미 / 여러 개 코어가 있는 시스템에서는 각각의 코어에 스레드를 할당함으로써 병렬로 작동 가능

멀티코어 프로그래밍 challenges area : identifying tasks (to find) / balance (equal work equal value) / data splitting (run on separate core) / data dependency (must be examined) / testing and debugging

Amdahl's Law (암달의 법칙) : 코어 추가시 성능개선은 직렬 / 병렬 요소를 모두 갖는다 /speed up <= 1 / S + (1-s)/N / serial portion이 존재하는 한 아무리 병렬로 컴퓨팅성능을 향상시켜도 한계가 있다

User thread : 스레드 라이브러리에서 스레드 생성 및 제거 위한 코드 융지 / 스레드간 데이터 메시지 전송 / 커널모드 권한을 요구하지 않는다 / OS에서만 돌아감 / 생성과 관리가 빠르다 / 하나의 유저스레드가 blocking operation 수행하면 전체 프로세스가 막힘

Kernel thread : OS로부터 직접 supported and managed / 동일프로세스 여러 스레드가 다른 프로세서에 스케쥴 될 수 있다 / 하나의 스레드가 블락 되어도 동일프로세스 다른 스레드는 스케쥴될 수 있다 /

Many to One Model : 여러 개 유저 스레드가 하나의 커널스레드로 연결

One to One Model : 각각의 유저스레드를 단일 커널 스레드로 연결 / 컨커런시 향상 / 멀티프로세서에서 병렬로 동작

many to many : 여러 유저스레드가 그보다는 적거나 같은 수의 커널스레드로 다중전송 / parallelism은 아니다 왜냐면 커널은 한번에 하나의 스레드만 스케쥴하기 때문 /

Two-level Model : 다수 유저 스레드를 적거나 같은 수의 커널스레드로 다중 전송

chapter 5.

CPU 스케쥴러 : 1) running->waiting switch (I/O) / 2) running->ready switch (time slice expired) / 3) waiting -> ready (새로운 프로세스가 레디큐로 들어온 것) / 4) terminates (exit)

1),4) ready큐를 건들지 않는다. -> nonpreemptive / 2)3) 지금수행하는 프로세스 중단 preemptive

Disptcher : switching context 관장 (이전 수행하던 프로세스를 저장, 새 프로세스 리로드)

Scheduling Criteria : CPU Utilization, Throughput, Turnaround time, Waiting time, Response time

FCFS (First come First Served) : 레디큐에 들어온 순서대로 그냥 처리 (심플)

(Convoy effect: cpu burst time이 긴 프로세스 하나 처리한다고 다른 프로세스들이 기다리는 현상)

SJF (Shortest Job First) : cpu burst time 이 가장 짧은 프로세스부터 처리

Length of Next CPU burst : t번째 일어난 것 * a + t번째 일어날것으로예상했던 것 * (1-a)

Shortest-remaining time first : 남은 시간을 비교하여 짧은 것부터

Non-preemptive SJF :

Preemptive SJF :

Round Robin : 퀀텀 토막 토막

Priority Scheduling : 우선순위부터 (동일하면 Round Robin 2씩 적용 번갈아가면서)

Multilevel Queue Scheduling

Multilevel Feedback Queue Scheduling

Multiple Processor Scheduling: coarse grained(스레드마다 처리해야할 데이터 양이 다름) fine grained / load balancing과 processor affinity

Memory Stall: 프로세스가 메모리에 엑세스할 때 available 해지기 위해 많은 시간 소비

Rate-Monotonic Scheduling : period (deadline)을 기준으로 어떤 작업을 할건지 preemption 적용

Earliest Deadline First Scheduling : 그 시점 말고 바로 다음시점의 period를 보고 preemption