용어해설

Glossary

- **2도 릴레이션** binary relation 관계형 데이터베이스에서 2도의 관계
- **2차 기회 페이지 교체 전략** second-chance page-replacement strategy FIFO 페이지 교체 전략의 변형으로, 참조 비트와 FIFO 큐를 사용해 교체 대상을 결정한다. 가장 오래된 페이지의 참조 비트가 'off'면 그 페이지를 교체하고, 그렇지 않으면, 가장 오래된 페이지의 참조 비트를 'off'로 설정한 후 FIFO 큐의 꼬리 쪽으로 옮긴다. 만일 그 페이지의 참조 비트가 'on'이면, 비트를 'off'로 설정한 후(즉, 지금은 넘어가고 2차 기회를 주고) 참조 비트가 'off'인 페이지를 찾을 때까지 다음 페이지들을 탐색한다.
- **2차 저장소** secondary storage 다량의 데이터를 영구적으로 저장하는 메모리. 2차 저장소는 메모리 계층에서 메인 메모리보다 한 단계 낮은 계층에 있다. 컴퓨터 전원이 들어온 후 정보들은 2차 저장소와 메인 메모리 사이를 이동해 프로세서가 프로그램 명령어와 데이터들이 참조할 수 있게 한다. 하드 디스크는 가장 흔히 사용하는 2차 저장소다.
- **3DES** '삼중 DES' 참고
- **3도 릴레이션** ternary relation 3도의 관계
- **AFRAID** A Frequently Redundant Array of Independent Disks **(RAID)** RAID 구현으로, 시스템 부하가 적을 때까지 패리티 데이터 생성을 연기한다. 시스템이 적은 양의 쓰기 작업을 많이 수행할 때 성능을 향상할 수 있다.
- **AGP** Accelerated Graphics Port 그래픽 장치들을 연결하는 데 많이 사용하는 버스 아키텍처. AGP는 대체로 초당 260MB 대역폭을 지원한다.
- **ARPA** Advanced Research Projects Agency 인터넷의 기초를 세운 미 국방부 산하의 정부 기관으로, 현재는 DARPA Defense Advanced Research Projects Agency라고 한다.
- **ARPAnet** 연구자들이 컴퓨터를 네트워크에 연결하도록 해준 인터넷의 전신. 가장 큰 장점은 훗날 이메일로 알려진 방법을 통해 쉽고 빠르게 통신할 수 있다는 점이다.
- **bio 구조체** bio structure **(리눅스)** 입출력 요청을 페이지에 맵핑함으로써 블록 입출력 연산을 간략화해주는 구조체
- **BSD 유닉스** BSD(Berkeley Software Distribution) UNIX 버클리 대학교에서 빌 조이가 이끈 팀에서 개발하고 발표한 유닉스 수정 버전. BSD 유닉스는 몇 가지 유닉스 변종의 전신이 되었다.
- **C** 데니스 리치가 개발한 절차적 프로그래밍 언어로, 유닉스 시스템을 만드는 데 사용되었다.
- **C 스레드** C-thread (매킨토시 OS X에서 작성한) 마크 마이크로커널에서 지원하는 스레드
- **C#** 마이크로소프트에서 개발된 객체지향 언어로, 닷넷 라이브러리에 접근할 수 있게 해준다.
- **C++** 비아네 스트롭스트룹이 개발했으며, C 언어를 객체지향으로 확장한 언어
- **C-LOOK 디스크 스케줄링** C-LOOK(Circular LOOK) disk scheduling 암을 한 방향으로 움직이는 디스크 스케줄링 전략. 최소 탐색 시간 기반으로 요청들을 서비스한다. 현재 방향에 더는 요청이 없을 때는 읽기/쓰기 헤드가 현재 위치 반대쪽 실린더에서 가장 가까운 요청으로 이동해 다음 번 sweep을 시작한다. C-LOOK 정책은 LOOK에 비해 반응 시간의 변량이 적고, LOOK보다는 적지만 처리량도 많다는 것이 특징이다.
- **C-SCAN 디스크 스케줄링** C-SCAN(Circular SCAN) disk scheduling 암을 한 방향으로 이동하며 최소 탐색 시간 기준

으로 요청들을 서비스하는 디스크 스케줄링 전략. 암이 sweep을 끝내면 요청들을 서비스하지 않고 현재 위치의 반대쪽 실린더로 이동한 후 다시 안쪽 방향으로의 sweep을 시작한다. C-SCAN은 가장 안쪽이나 바깥쪽 실린더에 대한 차별을 완화해 반응 시간의 변량을 줄이면서도 높은 수준의 처리량을 제공한다.

- **CD** Compact Disk 디지털 저장 매체로, 평평한 표면에 미세한 홀을 내어 데이터를 표현한다.
- **CMS 배치 기능** CMS Batch Facility **(VM)** 사용자들이 더 긴 작업을 개별적인 가상 머신에서 수행할 수 있게 함으로써 대화식 작업을 지속할 수 있게 해주는 VM 구성 요소
- **COM** Component Object Model 마이크로소프트에서 개발한 소프트웨어 아키텍처로, 표준 객체 인터페이스를 통해 다양한 컴포넌트 사이의 상호 운용성을 허용한다.
- **CORBA** Common Object Request Broker Architecture 1990년대 초기에 OMG Object Management Group이 고안했다. CORBA는 분산 시스템 아키텍처의 표준 명세며, 널리 수용되고 있다.
- **CP/CMS** 1960년대에 IBM에서 개발한 시분할 운영체제
- **CTSS** 1960년대에 MIT에서 개발한 시분할 운영체제
- **DCOM** Distributed Component Object Model 마이크로소프트의 COM을 분산 시스템용으로 확장한 것
- **DDR** Double Data Rate 클록 사이클당 메모리 전송을 두 번 수행해, 프론트사이드 버스가 자신의 클록 속도의 두 배로 작동할 수 있게 해주는 칩셋 기능. 이 기능은 시스템의 칩셋과 RAM에서 지원해야 한다.
- **DMA 메모리** DMA memory **(리눅스)** 물리 메모리 영역 중 0~16MB 사이로, 대체로 커널 부트스트랩핑 코드와 레거시 DMA 장치들을 위해 예약되는 것이 보통이다.
- **EBCDIC** Extended Binary-Coded Decimal Interchange Code 8비트 문자 집합으로, 메인프레임 컴퓨터 시스템, 특히 IBM에서 개발한 시스템의 데이터를 나타낸다.
- **ext2 아이노드** ext2 inode **(리눅스)** ext2 파일 시스템에서 단일 파일이나 디렉토리를 위해 파일 크기, 파일의 데이터 블록 위치 등의 정보를 담은 구조
- **ext2fs** second extended file system **(리눅스)** 유명한 아이노드 기반의 리눅스 파일 시스템으로 작은 파일에 빠르게 접근할 수 있게 해주고 큰 파일 크기를 지원한다.
- **FAT 파일 시스템** FAT file system 마이크로소프트에서 개발했으며, 테이블을 사용해 불연속적 파일 할당을 구현한 파일 시스템
- **FIFO (리눅스)** 서로 무관한 두 프로세스들이 생산자/소비자 관계를 통해 통신할 수 있게 해주는 이름 있는 파이프. 페이지 크기의 버퍼를 사용한다.
- **FIFO 브로드캐스트** FIFO broadcast 두 메시지가 한 프로세스에서 다른 프로세스로 전송될 때, 먼저 보낸 메시지가 먼저 도착하는 것을 보장하는 것
- **FIFO 이상 현상** FIFO Anomaly FIFO 페이지 교체 전략에서 프로세스에 할당되는 페이지 프레임 수가 증가할 때 오히려 페이지 폴트 횟수가 많아지는 현상. 일반적으로는 페이지 프레임을 많이 가질수록 페이지 폴트가 적어지므로, 이를 '이상 현상'이라고 한다.

- **FSCAN 디스크 스케줄링** FSCAN disk scheduling SCAN 디스크 스케줄링 중에서도 특정 sweep이 시작될 때 이미 와서 대기한 요청들만 서비스해주는 것. 'F'는 특정 시간에 요청 큐를 'freezing' 하는 것을 의미한다. sweep이 시작된 후에 도착하는 요청들은 그룹화되고 최적의 순서로 정렬되어 다음 번 sweep을 위해 준비된다.
- **GNU** GNU's Not UNIX 1980년대에 리처드 스톨만이 오픈 소스 운영체제를 만들 목적으로 시작한 프로젝트로, 유닉스의 특징과 기능을 사용한다.
- **GPL** General Public License 오픈 소스 소프트웨어 라이선스로, 이 라이선스를 따르는 소프트웨어는 완성된 소스 코드를 포함해 배포하며, 원본 코드를 수정할 수 있고, GPL 라이선스를 부여해야 함을 명시하고 있다. 사용자들은 소스 코드를 자유롭게 수정하고 재배포할 수 있다.
- **HTML** HyperText Markup Language 웹 페이지의 내용을 구성하고, 다른 페이지로 링크를 제공하는 방법을 명시하는 언어
- **HTTP** HyperText Transfer Protocol 클라이언트와 서버 사이에서 HTML 문서와 기타 데이터를 전송하기 위한 네트워크 프로토콜로, 월드 와이드 웹의 핵심 프로토콜이다.
- **IBSYS** IBM 7090 메인프레임용 운영체제
- **IEEE 1394 포트** IEEE 1394 port 흔히 사용하는 직렬 포트로, 초당 800MB까지 속도를 낼 수 있다. 때로는 장치에 전력을 공급하고, 시스템이 실행하는 동안에도 장치를 추가, 삭제할 수 있게 한다. 이러한 포트를 흔히 (애플에서는) FireWire 혹은 (소니에서는) iLink라고 한다.
- **IP 스푸핑** IP spoofing 공격자가 인증된 사용자나 호스트의 IP 주소를 시뮬레이션해 자원에 대한 접근 허가를 얻어내는 공격
- **ksoftirqd (리눅스)** softirq 부하기 높을 때 소프트웨어 인터럽트 처리기를 스케줄링하는 데몬
- **kswapd (리눅스)** 페이지를 디스크에 스왑하는 데몬
- **LOOK 디스크 스케줄링** LOOK disk scheduling SCAN 디스크 스케줄링 전략의 변형으로, 현재 sweep을 끝까지 미리 봄으로써 다음 서비스할 요청을 결정한다. 현재 방향에 더는 요청이 없을 때는 선호 방향을 바꾸어 다음 sweep을 시작하고, 큐에 있는 요청에 대응하는 실린더를 지날 때 멈춘다. 이 전략은 불필요한 탐색 연산을 줄여준다.
- **LSB** Linux Standard Base 커널 버전(그리고 배포판) 사이에서 응용 프로그램의 이식성을 높이기 위해 표준 리눅스 인터페이스를 지정하는 프로젝트
- **MD5 메시지 다이제스트 알고리즘** MD5 message digest algorithm MIT의 로널드 리베스트 교수가 개발한 해시 알고리즘으로, 디지털 서명을 구현하는 데 널리 사용되었다.
- **MS-DOS** 최초의 IBM 개인용 컴퓨터를 위한 운영체제로, 마이크로컴퓨터들과 호환성이 있다.
- ***m*대*n* 스레드 맵핑** *m*-to-*n* thread mapping '다대다 스레드 맵핑' 참고
- ***N*-Step SCAN 디스크 스케줄링** *N*-Step SCAN disk scheduling 큐에 있는 처음 n개의 요청을 서비스하는 디스크 스케줄링 전략. 훑기 완료될 때 다음 n개의 요청을 서비스한다. 도착하는 요청들은 요청 큐 끝에 놓인다. *N*-Step SCAN은 처리량이 높고 평균 반응 시간이 짧으며 반응 시간 변량도 SSTF와 SCAN보다 적다.

- **notify** 모니터의 대기 집합에 있는 스레드 하나를 깨울 수 있는 자바 메소드. 어떤 스레드가 깨어나는지는 자바 가상 머신 구현에 따라 다르다.
- **notifyAll** 모니터의 대기 집합과 엔트리 집합에 있는 모든 스레드를 깨우는 자바 메소드. 이 메소드는 대기하는 스레드들이 무기한 연기되는 일을 방지해주지만, notify 메소드보다 많은 오버헤드가 발생한다.
- ***n*도 릴레이션** *n*-ary relation *n*도의 관계
- **ODBC** Open DataBase Connectivity 응용 프로그램이 다른 인터페이스를 사용하는 다양한 데이터베이스에 접근할 수 있게 해주는 미들웨어 프로토콜. ODBC 드라이버는 데이터베이스 연결을 처리하고 응용 프로그램이 요청한 정보들을 조회해준다. 이로써 응용 프로그램은 특정 데이터베이스를 위한 코드를 작성하지 않아도 된다.
- **OpenBSD** 보안을 최우선으로 한 BSD 유닉스 시스템
- **ORB** Object Request Broker CORBA 클라이언트와 서버에 모두 있는 구성 요소로, 시스템 간의 통신을 주도하는 것을 담당한다.
- **OS/360** IBM System/360 메인프레임용 운영체제. OS/360은 두 가지 주요 옵션, 즉 MFT Multiprogramming with a Fixed number of Tasks와 MVT Multiprogramming with a Variable number of Tasks를 갖는다. OS/360-MVT는 MVS로 발전했고, 이는 현재의 IBM 메인프레임 운영체제 z/OS의 선조가 되었다.
- **OSF** Open Software Foundation 유닉스 개발자 연합으로, AT&T와 썬의 솔라리스와 경쟁하며 비독점적인 유닉스 버전인 OSF/1을 개발했다. OSF와 AT&T/썬 연합은 유닉스 전쟁의 두 축이었다.
- **OSF/1** OSF에서 솔라리스와 경쟁하려고 만든 유닉스 버전
- ***P* 연산** *P* operation 세마포어에 수행되는 연산으로, 세마포어 값이 0이면, *P* 연산은 자신을 호출하는 스레드를 블록한다. 세마포어 값이 0보다 크면, 값을 1만큼 감소시키고 자신을 호출하는 스레드가 작업을 진행할 수 있게 한다.
- **PCI 버스** PCI(Peripheral Components Interconnect) bus 네트워크 카드나 사운드 카드 등의 주변 장치를 시스템의 다른 부분에 연결하는 데 많이 사용하는 버스. PCI는 32비트 혹은 64비트 버스 인터페이스를 제공하고, 초당 533MB까지 전송률을 지원한다.
- **PGP** Pretty Good Privacy 공개키 암호화 시스템으로, 이 메인 메시지와 파일을 암호화하기 위해주로 사용되었고, 1991년에 짐머만이 고안했다.
- **POSIX** Portable Operating Systems Interface 초기 유닉스 운영체제에 기반을 둔 API
- **proc 파일 시스템** procfs, proc file system **(리눅스)** 커널에 직접 만들어진 파일 시스템으로, 메모리 활용도나 시스템의 실행 시간 등 커널과 프로세스에 관한 상태 정보를 실시간으로 제공한다.
- **Pthread** POSIX 1003.1c thread POSIX 1003.1c 표준을 준수하는 스레드
- **RAID** Redundant Array of Independent Disks 디스크 배열을 사용해 디스크 전송률을 증가시키고 장애 내구성을 제공하는 기술

- **RAID 수준 0** RAID level 0 스트라이프된 디스크 배열을 사용하며 중복성은 없는 RAID 시스템. 수준 0은 장애 내구성이 없다. 디스크 하나가 고장 나면 해당 디스크에 의존하는 배열 내 모든 데이터를 손실한다. 배열의 스트립 크기에 따라, 한 디스크가 손실되어도 배열의 모든 데이터가 손실된다. RAID 0은 장애 내구성은 없지만 저장소 오버헤드를 초래하지 않고, 슈퍼컴퓨터와 같이 장애 내구성보다 성능이 더 중요한 시스템에서 구현된다.
- **RAID 수준 1** RAID level 1 중복을 위해 디스크 미러링(섀도잉이라고도 한다)을 사용하는 RAID 시스템으로, 배열의 각 디스크는 사본을 갖는다. 스트라이프는 수준 1에서는 구현되지 않으며, 따라서 하드웨어 복잡도와 시스템 성능 모두를 감소시킨다. 장애 내구성은 가장 높지만 디스크 배열의 절반만이 고유한 데이터를 저장하는 데 사용되므로 비용이 증가한다. 수준 1 배열은 데이터베이스 시스템과 같이 비용보다 신뢰성이 중요한 시스템에서 구현된다.
- **RAID 수준 2** RAID level 2 비트 수준에서 스트라이프된 RAID 시스템으로, 각 스트립은 한 비트를 저장한다. 수준 2 배열은 미러링을 통해 저장소 오버헤드를 줄이려고 고안되었다. 각 데이터 항목의 복사본을 유지하는 대신, RAID 수준 2는 해밍 오류 정정 코드 Hamming ECC를 사용해서 패리티 정보를 저장해 시스템이 오류 두 개를 탐지하고 오류 한 개를 정정하고 스트라이프에서 오류의 위치를 파악할 수 있게 해준다. 해밍 ECC 코드의 크기 및 이로 인한 패리티 디스크의 크기는 데이터 디스크 수의 로그(밑수 2) 수준으로 증가한다. 따라서 수준 2 배열이 많은 수의 디스크를 갖는 경우 수준 1 배열보다 상당히 적은 저장소 오버헤드를 일으킨다.
- **RAID 수준 3** RAID level 3 RAID 시스템은 비트나 바이트 수준에서 데이터를 스트라이핑한다. RAID 3은 배타적 논리합 오류 정정 코드 XOR ECC를 사용한다. 이는 논리적 XOR 연산을 사용해 패리티 정보를 생성한다. XOR ECC는 패리티 정보를 유지하기 위해 배열 크기에 상관없이 한 디스크만 사용한다. 시스템은 패리티 비트를 사용해 한 디스크가 고장 난 것을 복구할 수 있다. 패리티 확인 때문에 RAID 수준 3에서 읽기와 쓰기 요청을 위해서는 전체 배열을 접근해야 한다. RAID 수준 2와 유사하게, 이는 큰 파일을 읽고 쓸 때, 높은 전송률을 내지만, 한 번에 한 요청만 서비스할 수 있다.
- **RAID 수준 4** RAID level 4 고정 크기 블록을 사용해 스트라이프된 RAID 시스템. XOR ECC를 사용해 패리티 데이터를 생성하고 한 패리티 디스크에 저장한다. 수준 4 배열은 큰 단위 스트라이핑을 가능하게 하므로 패리티가 각 읽기 작업에 대해 결정되지 않는다면, 시스템은 다중의 읽기 요청을 동시에 서비스할 수 있다. 그러나 쓰기 요청을 서비스할 때는 패리티 정보를 갱신해 디스크 고장 시 손실되는 데이터가 없도록 해야 한다. 이는 쓰기 요청은 한 번에 하나씩 서비스되어야 하며 따라서 쓰기 병목 현상이 발생함을 의미한다.
- **RAID 수준 5** RAID level 5 블록 수준에서 스트라이프된 RAID 시스템으로, XOR ECC를 사용해 패리티를 생성하지만, 패리티 블록은 디스크 배열 전체에 흩어져 있다. 패리티 블록이 분산되어 있으므로, 다중 패리티 스트립들이 동시에 접근될 수 있고, 많은 요청이 있을 시 쓰기 병목 현상을 없애준다. 수준 5는 수준 2~4보다 성능을 향상하지만, 구현하기 복잡하고 비용이 많이 든다. 수준 5 배열은 범용 배열로 인식되고 흔히 파일이나 애플리케이션 서버, ERP, 기타 비즈니스 시스템에서 찾아볼 수 있다.
- **RAID 컨트롤러** RAID controller 특수 목적 하드웨어로, 파일을 스트립으로 나누거나 스트립들로부터 파일 생성, 스트립의 위치 파악, 디스크 배열의 장애 내구성 메커니즘 구현 등의 연산을 효율적으로 수행해준다.
- **RAMAC** Random Access Method of Accounting and Control IBM에서 생산한 최초의 상용 하드 드라이브
- **ramfs (리눅스)** 메인 메모리 영역 중 블록 장치처럼 다루어지는 영역. ramfs 파일 시스템은 사용 전에 포맷해야 한다.

- **RMI의 스텁/스켈레톤 계층** stub/skeleton layer in RMI RPC의 클라이언트와 서버 스텁과 비슷한 파라미터 마셜링 구조를 가진 계층
- **RSA** 유명한 공개키 알고리즘으로, 1977년에 MIT의 로널드 리베스트, 아디 샤미르, 레오나르도 애들먼이 개발했다.
- **SCAN 디스크 스케줄링** SCAN disk scheduling 선호 방향에서 최소 탐색 거리를 요구하는 요청을 서비스함으로써 SSTF에 비해 불공평함과 반응 시간의 변량을 줄이는 디스크 스케줄링 전략. SCAN은 처리량이 많고 평균 반응 시간이 높다는 점에서 SSTF와 아주 비슷하지만, SCAN은 주어진 방향의 모든 요청들이 반대 방향보다 먼저 서비스되는 것을 보장하므로, SSTF보다 반응 시간의 변량이 적다. 이 전략은 엘리베이터 알고리즘이라고도 한다.
- **SCSI** Small Computer Systems Interface 여러 장치를 지원하고 고속 연결을 지원하기 위해 설계된 인터페이스. SCSI 인터페이스는 IDE 인터페이스보다 많은 장치를 지원하며, 애플 시스템과 많은 주변 장치를 갖춘 컴퓨터 시스템에서 많이 사용한다.
- **seqlock (리눅스)** 스핀 락을 시퀀스 카운터와 병합한 상호 배제 구조. seqlock은 즉각적, 배타적으로 데이터에 접근해야 하는 인터럽트 처리기에서 사용한다.
- **softirq (리눅스)** 소프트웨어 인터럽트 처리기로, 재진입 코드면서 직렬화(순차화)되지 않는다. 따라서 여러 프로세서에서 동시에 실행할 수 있다.
- **Solo** 페르 브린치 한센이 만든 작은 운영체제. 한센은 이 운영체제를 사용해 프로그램 실패에 안전한 fail-safe 병행 프로그래밍을 증명해보였다.
- **SUS** Single Unix Specification 오픈 그룹 Open Group에서 만든 명세로, 운영체제들은 이 명세를 준수해야 유닉스 상표를 보여줄 권한을 얻을 수 있다(http://www.unix.org/version3/overview.html) 참조
- **synchronized** 객체 내부의 코드에 상호 배제 접근을 하도록 만드는 자바 키워드
- **tasklet (리눅스)** 소프트웨어 인터럽트 처리기로, 여러 프로세서에서 동시에 실행될 수 없는 것들이다. tasklet은 인터럽트 처리기 중 재진입 방식이 아닌 하단 bottom half을 실행하는 데 사용된다.
- **TCP/IP** Transmission Control Protocol/Internet Protocol 인터넷에서 네트워킹의 프레임워크를 형성하는 프로토콜
- **THE 멀티프로그래밍 시스템** THE multiprogramming system 최초의 계층화된 운영체제 아키텍처로, 다익스트라가 개발했다.
- **USB** Universal Serial Bus 직렬 버스 인터페이스로, 초당 480Mbit에 이르는 데이터를 전송할 수 있다. 또한 장치에 전원을 공급하고, 장치의 핫 스와핑 기능을 지원한다.
- ***V* 연산** *V* operation 세마포어를 대기하는 스레드가 하나도 없을 경우에는 세마포어 변수 값을 증가시키는 연산. 대기하는 스레드가 있을 경우에는 *V* 연산이 그 중 한 스레드를 깨운다.
- **wait (세마포어)** 세마포어 변수 값이 0이면, wait 연산은 자신을 호출하는 스레드를 블록시킨다. 만일 변수 값이 0보다 크면, 이 연산은 변수 값을 1 감소시키고 호출하는 스레드가 진행할 수 있게 한다. wait 연산을 *P* 연산이라고도 한다.

- **WEP** Wired Equivalent Privacy 무선 보안 프로토콜로, 전송되는 데이터를 암호화하고 허가되지 않은 사용자가 무선 네트워크에 접근하는 것을 방지한다.
- **Win32 스레드** Win32 thread 마이크로소프트 32비트 윈도우 계열 운영체제에서 지원하는 스레드
- **WORM 매체** WORM(Write-Once, Read-Many) medium 한 번만 수정할 수 있는 저장 매체. 한 번 기록한 내용을 여러 번 읽을 수 있다.
- **WPA** Wi-Fi Protected Access 무선 보안 프로토콜로, WEP보다 개선된 암호화 방법을 제공하고 사용자 인증을 가능케 함으로써 WEP를 대체하기 위해 개발되었다.
- **z/OS** zSeries 메인프레임과 최근 MVS 버전을 위한 IBM 운영체제
- **가변 파티션 멀티프로그래밍** variable-partition multiprogramming 시스템에 유입되는 작업에 정확히 그 크기에 맞는 파티션을 할당하는 방법
- **가비지 컬렉션** garbage collection '메모리 압축' 참고
- **가상 머신** VM, Virtual Machine 컴퓨터 시스템의 기능을 소프트웨어적으로 구현한 프로그램. 가상 머신은 자신이 실행되는 물리적인 시스템과 직접 호환되지 않는 응용 프로그램을 실행할 수 있게 해준다. 사용자는 '가상 머신'이 아닌 하부에 있는 실제 시스템이라고 여긴다.
- **가상 머신 운영체제** VM(Virtual Machine)operating system 초기 가상 머신 운영체제 중 하나로, 1960년대에 IBM에서 개발하여 오늘날까지 널리 사용되고 있다. 최근 버전은 z/VM이다.
- **가상 메모리** virtual memory 프로그램이 실제로 할당받은 메인 메모리 용량보다 많은 메모리를 사용할 수 있게 하는 운영체제의 기능으로, 가상 메모리 시스템은 프로그래머가 메모리 관리에 신경 쓰지 않고, 응용 프로그램 로직에만 집중할 수 있게 해준다.
- **가상 메모리 영역** virtual memory area **(리눅스)** 프로세스의 가상 주소 공간의 연속적인 영역을 묘사해주는 구조로, 커널이 이 영역을 한 단위로 간주해 연산을 수행할 수 있게 해준다.
- **가상 사설망** VPN, Virtual Private Network 공중 통신 회선을 사용해 사설 네트워크에 원격 사용자를 안전하게 연결하는 기술. VPN은 흔히 IPSec를 사용해 구현한다.
- **가상 주소** virtual address 프로세스가 가상 메모리 시스템에서 접근하는 주소. 가상 주소는 실행 시에 동적으로 실제 주소로 변환된다.
- **가상 주소 공간** virtual address space 프로세스가 참조할 수 있는 메모리 주소 집합. 가상 주소 공간은 프로세스가 물리적으로 사용 가능한 메모리보다 많은 주소를 참조할 수 있게 해준다.
- **가상 파일 시스템** virtual file system **(리눅스)** 사용자에게 다양한 이기종 파일 시스템에 있는 파일과 디렉토리에 대한 공통의 뷰를 제공하는 인터페이스
- **가장 신뢰받는 사용자 상태** most trusted user status '커널 모드' 참고
- **간격 타이머** interval timer 주기적으로 인터럽트를 발생해 운영체제가 실행을 멈추게 하는 하드웨어. 이로써 악의적이거나 잘못 작성된 프로세스가 프로세서를 독점하는 일을 방지한다.

- **간접 블록** indirect block 아이노드 기반 파일 시스템에서 데이터 블록을 가리키는 포인터를 담은 인덱스 블록
- **간접 포인터** indirect pointer 아이노드 포인터들의 블록을 가리키고 있는 아이노드 포인터
- **강도 높은 자원 관리** intensive resource management 전반적인 자원 활용도를 증가시키기 위해 상당량의 자원을 투자해 다른 자원들을 관리하는 방법
- **강제 접근 제어** MAC, Mandatory Access Control 모든 주체와 객체의 접근 권한 정책을 미리 정의하도록 요구하는 접근 제어 모델
- **개인키** private key 공개키 암호화에서 소유자만 알고 있어야 하는 키. 대응하는 공개키가 메시지를 암호화하면, 그 비밀키만 이를 복호화할 수 있어야 한다.
- **객체** object 재사용 가능한 소프트웨어 구성 요소로, 실세계 객체들의 속성과 동작을 본떠서 정의한다.
- **객체 직렬화** object serialization 객체들이 바이트 스트림으로 인코딩되어 한 주소 공간에서 다른 주소 공간으로 전송될 수 있게 하는 일
- **객체지향 운영체제** OOOS, Object-Oriented Operating System 구성 요소와 자원을 객체로 표현하는 운영체제. 상속이나 인터페이스 같은 객체지향 개념은 모듈화된 운영체제를 가능하게 해주며, 기존 기술로 만든 운영체제보다 유지보수와 확장이 쉽다. 많은 운영체제가 객체를 사용하지만 전적으로 객체지향 언어를 사용해 만드는 운영체제는 거의 없다.
- **객체지향 프로그래밍** OOP, Object-Oriented Programming 프로그래머들이 복잡한 소프트웨어 시스템을 '객체'라는 컴포넌트들을 재사용해 더 빠르고 쉽게 만들 수 있는 프로그래밍 스타일. 객체는 해당 타입의 속성과 동작 방식을 정의하는 '클래스'라는 설계도에서 만들어진다.
- **갱신** update **(파일)** 파일의 기존 데이터 항목을 수정하는 연산
- **갱신 이미지** update image **(RAID)** 읽기-수정-쓰기 사이클 read-modify-write cycle을 수행하는 대신 새 패리티와 이전 패리티의 차이만 메모리에 저장함으로써 패리티 계산에 드는 시간을 줄이는 방법
- **견고한 운영체제** robust operating system 장애 내구성이 있고 신뢰성이 높은 운영체제로, 응용 프로그램이나 하드웨어의 갑작스런 오류 때문에 시스템 전체가 실패하지 않으며, 실패하더라도 우아한 퇴보를 한다. 이러한 운영체제는 하드웨어가 실패하지 않는 한 각 응용 프로그램에 적절한 서비스를 제공할 수 있다.
- **결합** join 스레드 연산의 하나로, 이 연산을 호출하는 스레드는 결합하는 스레드가 종료할 때까지 블록되어 대기한다. 주 스레드는 흔히 자신이 생성하는 스레드를 결합해 프로세스의 모든 스레드가 마칠 때까지 프로세스가 종료하지 않게 한다.
- **경계 레지스터** boundary register 단일 사용자 운영체제용 레지스터. 사용자 메모리 영역을 커널 메모리 영역에서 분리함으로써 메모리를 보호하는 데 사용한다.
- **경계 레지스터** bounds register 프로세스가 접근할 수 있는 메모리 주소의 범위 정보를 저장하는 레지스터
- **경계 버퍼** bounded buffer '원형 버퍼' 참고. 원형 버퍼는 생산자/소비자 관계에서 고정 크기 공유 메모리로, 생산자가 생산한 여러 값을 저장한다. 만일 생산자가 이따금씩 소비자보다 빠른 속도로 값을 생산할 경우, 원형 버퍼

는 단일 값을 저장하는 버퍼에 비해 생산자가 소비자를 대기해야 하는 시간을 줄여준다. 소비자가 일시적으로 빠른 경우에도 소비자가 생산자의 생산을 대기하는 시간을 줄여줄 수 있다.

- **경량 프로세스** LWP, LightWeight Process 프로그램 명령어의 단일 스레드(실행 스레드 혹은 프로그램 흐름 제어 흐름이라고도 한다) 스레드를 경량이라고 하는 이유는 동일 프로세스에 있는 다른 프로세스들과 주소 공간을 공유하기 때문이다.
- **경로명** pathname 파일이나 디렉토리를 논리적인 이름으로 식별하는 문자열. 디렉토리는 / 또는 \ 등의 구분자로 분리한다. 절대 경로명은 파일이나 디렉토리 위치를 루트 디렉토리에서부터 시작해 나타낸다. 상대 경로는 파일이나 디렉토리의 위치를 현재 작업 디렉토리에서부터 나타낸다.
- **경험적 방법** heuristic 복잡한 문제를 접근할 때 어림짐작이나 근사치, 경험치에 근거한 임기응변적인 방법 등을 통해 실행 오버헤드는 적게 들이면서도 대체로 좋은 결과를 내는 방법
- **경험적 스캐닝** heuristic scanning 프로그램의 동작 패턴을 보고 바이러스인지 탐지해내는 안티바이러스 기술
- **계산 중심** compute-bound '프로세서 중심' 참고
- **계층적 교착 상태 탐지** hierarchical deadlock detection 분산 교착 상태 탐지 전략으로, 시스템에 있는 각 사이트를 트리 구조 내의 노드로 정렬한다. 단말 노드를 제외한 각 노드는 모든 종속적인 노드들의 자원 할당 정보를 수집한다.
- **계층적 생체 인식 검증** LBV, Layered Biometric Verification 사람 특징의 여러 측정치를 사용하는 인증 기술. 예를 들면, 사용자의 신원을 확인하기 위해 얼굴, 지문, 음성 등을 사용한다.
- **계층적 운영체제** layered operating system 유사한 구성 요소끼리 그룹화해 계층으로 분류한 운영체제로, 각 계층은 자신의 하위 계층에 작업을 요청하고, 결과를 자신의 상위 계층에 반환한다.
- **계층적 프로세스 구조** hierarchical process structure (특히 한) 부모 프로세스가 자식 프로세스를 생성할 때 프로세스 구성
- **계층 구조 파일 시스템** hierarchically structured file system 파일 시스템 구성 중 각 디렉토리가 여러 하위 디렉토리와 정확히 한 부모 디렉토리만 지니는 파일 시스템
- **고급 구성 및 전원 인터페이스** ACPI, Advanced Configuration and Power Interface 전원 관리 명세로, 많은 운영체제에서 지원한다. ACPI 지원 시스템은 작업 손실 없이 한 개 혹은 여러 장치의 전원을 끌 수 있다.
- **고급 암호화 표준** AES, Advanced Encryption Standard 대칭형 암호화 표준으로, 암호화 방법으로 레인달을 사용한다. AES는 더욱 강화된 보안을 제공하므로 데이터 암호화 표준 DES, Data Encryption Standard를 대체했다.
- **고급 언어** high-level language 사람이 인식하기 쉬운 영어와 유사한 구문과 식별자, 보편적인 수학 기호를 사용하는 프로그래밍 언어로, 어셈블리 언어보다 적은 문장을 사용해 많은 작업을 수행할 수 있다.
- **고도 병렬성** massive parallelism 여러 프로세서를 보유해 계산의 많은 부분을 병렬로 수행하는 시스템 속성
- **고수준 스케줄링** high-level scheduling 시스템이 자원을 위해 활발하게 경쟁할 작업을 결정하는 일

- **고장 투명성** failure transparency 분산 시스템이 장애 내구성을 제공해 클라이언트 측에서는 컴퓨터가 고장 난 것을 인식하지 못하도록 하는 방법
- **고정 파티션 멀티프로그래밍** fixed-partition multiprogramming 메인 메모리를 일정 수의 고정 크기 파티션으로 나누고 각 파티션이 한 작업을 보유하게 하는 메모리 구성
- **고정된 동기화** lockstep synchronization 비동기 스레드들이 엄격히 교대해가면서 코드를 수행하는 것
- **공간적 지역성** spatial locality 경험론적인 속성으로, 페이징 시스템에서 프로세스들이 페이지들의 일부분을 선호하는 경향이 있고, 이 페이지들은 해당 페이지의 가상 주소 공간 안에서 가까이 위치할 확률이 높다는 것이다. 예를 들어, 배열의 인덱스들을 순차적으로 접근하는 프로세스는 공간적 지역성을 보인다.
- **공개** public **(파일 접근 제어)** 시스템의 사용자 집단에 있는 누구든 접근할 수 있는 파일 접근 속성
- **공개키** public key 공개 암호 기법에서 소유자와 통신하려고 하는 모든 사용자가 사용할 수 있는 키. 공개키로 메시지를 암호화하면, 대응하는 비밀키를 소유한 사람만이 이를 복호화할 수 있다.
- **공개키 기반 구조** PKI, Public Key Infrastructure 공개키 암호 기법과 디지털 인증서, 인증기관을 통합해 트랜잭션 당사자를 인증하는 기술
- **공개키 암호 기법** public-key cryptography 비대칭 암호화 기술로, 역으로 연관된 두 키인 공개키와 비밀키를 사용한다. 송신자가 메시지를 안전하게 전송하려면 수신자의 공개키를 사용해 메시지를 암호화해야 한다. 그리고 수신자는 자신의 고유 비밀키를 사용해 해당 메시지를 복호화한다.
- **공유 라이브러리** shared library 여러 프로그램이 공유하는 함수 집합
- **공유 자원** shared resource 프로세스 하나 이상에서 접근할 수 있는 자원
- **공평 분배 그룹** fair share group 공평 분배 스케줄링 정책에서 프로세서 시간의 일부를 얻는 프로세스 그룹
- **공평 분배 스케줄링** FSS, Fair Share Scheduling AT&T 유닉스 시스템용으로 개발한 스케줄링 정책으로, 프로세스들을 그룹에 넣고 이 그룹들에 프로세서 시간의 일부분을 할당해준다.
- **공평성** fairness 스케줄링 알고리즘이 프로세스들을 공정하게 대우하는 속성
- **관계 모델** relational model Codd가 제안한 데이터 모델로, 대부분의 현대 데이터베이스 시스템의 근간이다.
- **관리 모드** executive mode 보호된 모드로, 프로세스가 사용자를 대신해 운영체제 명령어를 실행할 수 있다. 커널 모드라고도 한다.
- **교착 상태** deadlock 결코 일어나지 않을 이벤트를 대기하느라 스레드가 실행을 진행하지 못하는 상황
- **교착 상태 방지** deadlock prevention 네 가지 교착 상태 필요 조건 중 하나를 제거해 교착 상태를 방지하는 방법
- **교착 상태 복구** deadlock recovery 시스템에서 교착 상태를 제거하는 과정. 이를 위해서는 프로세스를 일시 정지(프로세스의 작업은 보존함)하거나 프로세스를 강제 종료(작업한 내용을 잃음)한 후 재시작한다.
- **교착 상태 비선점 필요조건** no-preemption necessary condition for deadlock 교착 상태 성립에 필요한 네 가지 조건 중 하나로, 프로세스로부터 강제로 자원을 빼앗을 수 없을 때만 교착 상태가 발생할 수 있음을 나타낸다.

- **교착 상태 상호 배제 필요조건** mutual exclusion necessary condition for deadlock 교착 상태 성립에 필요한 네 가지 조건 중 하나로, 프로세스가 자신의 자원을 배타적으로 사용하도록 요구할 수 있을 때만 교착 상태가 발생함을 나타낸다.
- **교착 상태 순환 대기 필요조건** circular-wait necessary condition for deadlock 교착 상태 성립에 필요한 네 가지 조건 중 하나로, 교착 상태가 존재한다면 프로세스 두 개 이상이 원형 체인을 이루면서 각 프로세스가 체인에서 다음 프로세스가 보유한 자원을 대기함을 나타낸다.
- **교착 상태 충분조건** sufficient conditions for deadlock 교착 상태가 발생하는 필요충분조건인 상호 배제 조건, 비선점 조건, 대기 조건, 순환 대기 조건을 가리킨다.
- **교착 상태 탐지** deadlock detection 시스템이 교착 상태에 빠져 있는지 여부를 판단하고, 탐지할 경우 교착 상태를 제거하는 일. 이 경우, 보통 작업 손실이 발생한다.
- **교착 상태 필요조건** necessary condition for deadlock 교착 상태가 발생하는 데 필요한 조건. 네 가지 필요조건은 상호 배제 조건, 비선점 조건, 대기 조건, 순환 대기 조건이다.
- **교착 상태 회피** deadlock avoidance 교착 상태 방지 알고리즘에 비해 제한을 덜 주어 성능을 높이면서도, 교착 상태가 실제로 발생하지 않도록 안전 상태를 유지하는 전략('다익스트라의 은행원 알고리즘' 참고)
- **교체 전략** replacement strategy **(메인 메모리)** 새로 들어오는 프로그램과 데이터를 수용하기 위해 기존 프로그램이나 데이터의 어느 부분을 교체할지 결정하는 데 사용하는 전략
- **구문 해석기** syntax analyzer '파서' 참고
- **구조적 프로그래밍** structured programming 분명하고 정확하며, 수정하기 쉬운 프로그램을 작성할 수 있는 잘 구조화된 프로그래밍 방법
- **구조화된 질의 언어** SQL, Structured Query Language 사용자들이 특정 속성을 가진 데이터 항목을 찾고, 테이블을 생성하고, 무결성 제약 조건을 지정하고, 데이터의 일관성을 관리하며 보안을 강화하는 데 사용하는 데이터베이스 언어
- **권한** privilege **(접근 권한)** 주체가 객체에 접근하는 방식
- **권한 부여** authorization **(보안 트랜잭션)** 성공적이고 안전한 트랜잭션을 위한 다섯 가지 기본 요소 중 하나로, 사용자의 자격 요건에 근거해 보호되는 자원에 대한 접근을 관리하는 방법을 다룬다.
- **규모 확장성** scalability 분산 시스템이 기존 응용 프로그램이나 사용자에게 영향을 미치지 않은 채 확장(예를 들면, 시스템에 머신 추가)할 수 있도록 해주는 속성
- **규모 확장성** scalability **(스케줄러)** 스케줄러가 시스템 부하가 클 때도 성능이 우아하게(점차적으로) 퇴보하도록 보장할 수 있는 특성
- **규모 확장성 있는 운영체제** scalable operating system 새로운 시스템 자원을 추가하기 쉬운 운영체제로, 사용자들의 필요에 맞게 멀티프로그래밍 정도 변화에 빨리 적응할 수 있다.
- **그래프 소거** graph reduction 자원 할당 그래프에서 작업을 완료할 수 있는 프로세스를 소거하는 방법. 자원 할당 그래프는 프로세스의 자원 요청을 모두 허용할 수 있을 경우 소거할 수 있으며, 이러한 프로세스는 실행을 완료

한 후 보유한 자원을 반납할 수 있다.

- **그래픽 사용자 인터페이스** GUI, Graphical User Interface 윈도우나 아이콘, 메뉴 등 사용자 친화적인 시각적 심볼을 제공해 운영체제에 쉽게 접근할 수 있게 한다.
- **그룹** group **(파일 접근 제어)** 동일한 파일 접근 권한을 가진 사용자들의 집합(예를 들면, 특정 프로젝트에서 일하는 구성원)
- **그룹 기술자** group descriptor **(리눅스)** 블록 그룹에 대한 정보를 기록하는 구조. 아이노드 할당 비트맵의 위치나 블록 할당 비트맵, 아이노드 테이블 등의 정보를 담고 있다.
- **기계어** machine language 컴퓨터 하드웨어 설계를 통해 정의하는 언어. 해당 컴퓨터에서 이해할 수 있는 언어다.
- **기능 동결** feature freeze **(리눅스)** 커널 개발 시에 새로운 기능을 커널에 추가하지 못하게 하는 상태로, 새로운 커널 발표를 준비하는 데 사용한다.
- **기본 접근 방법** basic access method 운영체제가 사용자의 입출력 요청에 즉시 반응하는 파일 접근 방법. 처리할 레코드의 순서를 예측할 수 없을 때, 특히 직접 접근 방식에서 사용한다.
- **기아** starvation 스레드가 전혀 발생하지 않을 수도 있는 이벤트를 대기하는 상황. '무기한 연기'라고도 한다.
- **기아 제한** starvation limit **(리눅스)** 우선순위가 높은 프로세스가 만료 리스트에 놓여 낮은 우선순위의 프로세스들이 무기한 연기되는 일을 방지하게 하는 시간
- **기준 레지스터** base register 프로세스가 참조할 수 있는 가장 낮은 메모리 주소를 담고 있는 레지스터
- **깨우기** wake 스레드 연산 중 하나로, 대상 스레드를 대기 상태에서 준비 상태가 되게 한다.
- **나이스 값** nice value **(리눅스)** 프로세스의 스케줄링 우선순위 척도. 낮은 나이스 값을 가진 프로세스들은 더 많은 프로세서 시간을 가질 수 있으므로, 시스템에 있는 다른 프로세스들에는 '덜 친절'하다.
- **내부 단편화** internal fragmentation 고정 파티션 멀티프로그래밍 시스템에서 프로세스의 데이터가 자신이 실행되는 파티션의 크기보다 작아서, 나머지 부분을 사용하지 못한 채 남기는 현상
- **네임스페이스** namespace 파일 시스템에서 식별할 수 있는 파일 집합
- **네트워크 기반 침입 탐지** network-based intrusion detection 네트워크 트래픽을 모니터링하는 침입 탐지 시스템으로, 서비스 거부 공격임을 나타낼 수 있는 비정상적인 동작 패턴이나 허가받지 않은 사용자가 네트워크를 접근하는 것을 탐지한다.
- **네트워크 운영체제** network operating system 원격에 있는 자원을 조작할 수 있는 운영체제로, 분산 시스템과 같이 자원의 위치를 감추지는 않는다.
- **네트워크 파일 시스템** NFS, Network File System 썬 마이크로시스템즈에서 구현한 파일 시스템으로, 각 컴퓨터를 서버/클라이언트로 동작할 수 있는 가상 파일 시스템을 가지고 있다.
- **넷필터 프레임워크** netfilter framework **(리눅스)** 커널 모듈이 패킷을 직접 검사하고 수정할 수 있게 해주는 메커니즘. 이는 패킷이 전송되기 전에 각 패킷의 원천 주소를 수정하는 방화벽 등의 응용 프로그램에 유용하다.

- **논리 레코드** logical record 소프트웨어에서 한 단위로 처리하는 데이터 집합
- **논리 뷰** logical view 파일을 저장하는 장치와 포맷, 시스템의 물리적 접근 방법 등을 감추는 파일 보기 관점
- **논리 블록** logical block '논리 레코드' 참고
- **논리 시계** logical clock 이벤트의 완전 순서를 만들기 위해 시스템에서 발생하는 각 이벤트에 타임스탬프를 부여하는 것
- **논리 주소 공간** logical address space **(IA-32 인텔 아키텍처)** 세그먼트에 담겨 있는 주소들의 집합
- **논리적 백업** logical backup 파일 데이터와 파일 시스템의 디렉토리 구조를 저장하는 백업 기술. 흔히 공통적인 압축 형태로 저장한다.
- **논리폭탄** logic bomb 특정 조건을 충족할 때 자신의 코드를 실행하는 바이러스
- **다대다 스레드 맵핑** many-to-many thread mapping 사용자 스레드 집합이 커널 스레드 집합에 할당되는 스레딩 모델로, 응용 프로그램은 커널 수준 스레드와, 스케줄러 활성화와 같은 사용자 수준 기능을 다 활용할 수 있다. 실제로는 사용자 스레드 수가 시스템의 커널 스레드 수 이상이 되게 하여 메모리 소비를 최소화한다.
- **다대일 스레드 맵핑** many-to-one thread mapping 프로세스의 모든 사용자 수준 스레드가 한 커널 스레드에 할당되는 스레딩 모델
- **다수준 페이징 시스템** multilevel paging system 시스템이 프로세스 페이지 테이블의 일부를 메인 메모리의 불연속적인 위치에 저장하고 프로세스가 활발히 사용하는 부분만 저장할 수 있게 해주는 기술. 다수준 페이지 테이블은 페이지 테이블의 계층 구조를 생성해 구현된다. 각 계층은 하위 계층의 테이블을 가리키는 포인터를 담은 테이블을 포함한다. 최하위 계층은 페이지-페이지 프레임 맵핑을 담은 테이블로 이루어져 있다. 이는 단일 수준 페이지 테이블에 비해 메모리 낭비를 줄여주지만, 대응하는 맵핑이 TLB에 없을 경우에는 추가적으로 메모리에 접근해야 한다는 오버헤드를 야기한다.
- **다수준 피드백 큐** multilevel feedback queue 동일한 우선순위를 가진 프로세스의 그룹을 동일한 라운드로빈 큐에 두는 프로세스 스케줄링 구조. 프로세서 중심 프로세스는 대체로 빠른 반응 시간을 요하지 않는 배치성일 때가 많기 때문에 낮은 우선순위 큐에 놓인다. 입출력 중심 프로세스는 입출력을 위해 시스템을 빨리 나오는 편이므로, 높은 우선순위 큐에 놓인다. 입출력 중심 프로세스는 반응 시간을 좋게 해줘야 하는 대화식 프로세스일 경우가 많다.
- **다익스트라 알고리즘** Dijkstra's Algorithm 가중치 그래프에서 최단 거리를 찾는 효율적인 알고리즘
- **다익스트라의 은행원 알고리즘** Dijkstra's Banker's Algorithm 교착 상태 회피 알고리즘으로, 시스템이 보유한 자원과 각 프로세스가 보유한 자원 그리고 해당 프로세스들이 실행 중에 요청할 최대 자원의 양에 근거해 자원 할당을 제어한다. 요청된 자원을 할당해도 시스템이 안전 상태를 유지할 때만 자원 할당을 허용한다('안전 상태'와 '불안전 상태' 참고).
- **다중 가상 공간** MVS, Multiple Virtual Spaces System/370용 IBM 운영체제로 16MB 가상 주소 공간을 몇 개든 가질 수 있다.
- **다형적 바이러스** polymorphic virus 알려진 바이러스 목록을 회피하려고 코드를 수정하는 바이러스. 예를 들면, 확산될 때 코드를 암호화하거나 대체, 삽입하는 등의 방법을 사용한다.

- **단일 사용자 연속 메모리 할당 시스템** single-user contiguous memory allocation system 프로그램들을 인접한 메모리 주소에 배치하고, 한 번에 한 프로그램에만 시스템 서비스를 주는 시스템
- **단일 수준 디렉토리 구조** single-level directory structure '평면 디렉토리 구조' 참고
- **단일 스트림 배치 처리 시스템** single-stream batch-processing system 보류 작업 큐 중에서 준비 상태 작업들을 가용 파티션에 배치하는 배치 처리 시스템
- **단편화** fragmentation **(메인 메모리)** 시스템이 메인 메모리의 여유 공간을 활용할 수 없게 되는 현상
- **단편화된 디스크** fragmented disk 파일들이 여러 번 추가 삭제됨에 따라 파일들을 불연속적인 블록에 저장하는 디스크. 이러한 디스크는 파일을 순차적으로 읽을 때 탐색 시간이 오래 걸린다. 이를 해결하려면 디스크 조각 모음을 해야 한다.
- **닫기** close **(파일)** 다시 열 때까지 파일에 더는 접근할 수 없게 하는 연산
- **대기 상태** waiting state 스레드 상태 중 하나로, 대기 상태 스레드는 깨우기나 통지 연산을 통해 준비 상태가 되기 전에는 실행될 수 없다.
- **대기 접근 방법** queued access method 사용자의 입출력 요청에 즉시 응답하지 않는 파일 접근 방법. 이 방법은 처리할 레코드의 순서를 예측할 수 있는 시스템에서 유용하며, 요청들을 정렬하여 접근 시간을 최소화할 수 있다.
- **대기 조건** wait-for condition 교착 상태 성립에 필요한 네 가지 조건 중 하나로, 프로세스가 다른 자원을 보유한 상태에서 다른 자원을 할당받으려고 대기할 수 있을 때만 교착 상태가 발생할 수 있음을 나타낸다.
- **대기 종료 교착 상태 방지 전략** wait-die deadlock prevention strategy 대기 조건을 거부해 교착 상태를 방지하는 방법. 생성된 시점을 기준으로 각 프로세스에 고유한 우선순위를 부여한다. 프로세스는 자신이 더 늦게 생성되었을 경우 다른 프로세스를 대기한다. 또한 자신이 대기하는 프로세스보다 먼저 생성된 프로세스는 종료된다.
- **대기 집합** wait set 자바에서 모니터에 대한 잠금을 다시 얻으려고 대기하고 있는 스레드의 집합
- **대기 큐** wait queue '대기 집합' 참고
- **대역폭** bandwidth 통신 회선에서 정보를 나를 수 있는 용량. 단위 시간에 전송되는 데이터양
- **대체 암호** substitution cipher 특정 문자들을 다른 문자로 바꾸는 암호화 기술. 예를 들어, 'a'를 'b'로, 'b'를 'c'로 대체하는 식이면, "security"라는 단어는 "tfdvsjuz"로 암호화된다.
- **대칭 암호 기법** symmetric cryptography '비밀키 암호 기법' 참고
- **대화식 모니터 시스템** CMS, Conversational Monitor System **(VM)** VM 구성 요소로, 대화식 응용 프로그램 개발 환경이다.
- **대화식 문제 제어 시스템** IPCS, Interactive Problem Control System **(VM)** VM 구성 요소로, VM 소프트웨어의 문제에 대해 온라인 분석과 수정을 제공한다.
- **대화식 사용자** interactive user 시스템이 작업을 진행하는 동안 시스템 앞에서 상호 작용하는 사용자를 말한다. 시스템 운영자에게 작업을 넘기고 결과만 받는 데 비해 작업을 진행하는 동안 시스템과 상호 작용할 수 있다.

- **대화식 운영체제** interactive operating system 사용자의 입력에 빠르게 반응하며 사용자와 상호 작용하는 운영체제
- **대화식 프로세스** interactive process 실행 시 사용자의 입력을 요구하는 프로세스
- **더티 비트** dirty bit 페이지가 수정되었는지를 나타내는 페이지 테이블 개체. 수정 비트 modified bit라고도 한다.
- **데드 상태** dead state 스레드가 자신의 작업을 마치거나 다른 방법으로 종료한 후에 진입하는 스레드 상태
- **데드라인 비율 단조 스케줄링** deadline rate-monotonic scheduling 실시간 시스템의 스케줄링 정책으로, 자신의 주기와 동일하지 않은 주기적인 프로세스의 데드라인을 충족한다.
- **데드라인 스케줄러** deadline scheduler **(리눅스)** 입출력 요청이 서비스되는 데드라인을 할당해 무기한 연기를 방지하는 디스크 스케줄링 알고리즘
- **데드라인 스케줄링** deadline scheduling 프로세스나 스레드가 제한된 시간 안에 완료되도록 스케줄링하는 일. 완료 데드라인이 임박함에 따라 프로세스나 스레드의 우선순위를 증가시켜야 한다.
- **데몬** daemon **(리눅스)** 시스템 서비스를 수행하기 위해 주기적으로 실행되는 프로세스
- **데스크톱 환경** desktop environment 윈도우 관리자 위에 있는 GUI 계층으로, 시스템 사용성을 높이기 위한 도구, 응용 프로그램, 기타 소프트웨어를 제공한다.
- **데이터 계층** data hierarchy 의미 있는 데이터를 추출하기 위해 그룹화할 비트 수를 분류하는 것. 비트 패턴, 바이트, 워드는 소수 비트를 묶고 하드웨어와 저수준 소프트웨어로 해석한다. 필드, 레코드, 파일 등은 다수의 비트로 구성되며 운영체제나 사용자 응용 프로그램으로 해석할 수 있다.
- **데이터 독립성** data independence 특정 파일 시스템의 구성과 접근 기술에 의존하지 않는 응용 프로그램 속성
- **데이터 마셜링** marshaling of data 클라이언트 스텁이 프로시저 인자와 반환 값을 패키지화해 네트워크에 전송할 수 있도록 하는 일
- **데이터 버스** data bus 주소 버스로 지정한 메모리 위치에서 데이터를 전송하는 버스
- **데이터 스트라이핑** data striping **(RAID)** RAID 시스템에서 연속적인 데이터를 고정 크기의 스트립으로 나누어 다른 디스크에 저장하는 기술. 이 기술은 여러 디스크를 사용해 데이터에 대한 요청을 서비스할 수 있게 해준다.
- **데이터 암호화 표준** DES, Data Encryption Standard 대칭 암호화 알고리즘으로 56비트 키를 사용해 데이터를 64비트 블록으로 암호화한다. DES는 여러 해 동안 미국 정부와 ANSI의 암호화 표준으로 지정되었다. 그러나 1990년대에 들어 컴퓨터 성능이 증가하면서 더는 안전하지 않게 되었고, DES 전문 크랙커 머신이 개발되어 몇 시간 만에 DES 암호를 알아내었다.
- **데이터 압축** data compression 반복적인 패턴을 짧은 비트열로 대체해 데이터 레코드의 크기를 줄이는 기술. 이 기술은 탐색 및 전송 시간을 줄여주지만, 데이터를 압축하고 해제하는 데 상당한 프로세서 시간을 소비한다.
- **데이터 영역** data region 한 프로세스의 주소 공간에서 (명령어와 대조적으로) 데이터를 저장하는 부분. 이 영역은 수정 가능하다.
- **데이터 의존적인 응용 프로그램** data-dependent application 특정 파일 시스템의 구성과 접근 기술에 의존하는 응용 프로그램

- **데이터 재생성** data regeneration **(RAID)** RAID 시스템에서 디스크 오류 등 때문에 손실된 데이터를 재건하는 일
- **데이터 정의 언어** DDL, Data Definition Language 데이터베이스의 데이터들의 구성을 지정하는 언어 유형
- **데이터 조작 언어** DML, Data Manipulation Language 데이터 수정을 가능하게 하는 언어 유형
- **데이터그램 소켓** datagram socket 데이터를 전송하기 위해 UDP 프로토콜을 사용하는 소켓
- **데이터베이스** database 중앙에서 통제하는 데이터 집합으로, 표준화된 포맷을 저장하고 데이터 간 논리 관계를 기반으로 검색한다. 데이터베이스는 데이터를 경로명이 아닌 내용에 따라 구성한다. 이는 정보의 중복성을 없애주는 경향이 있다.
- **데이터베이스 관리 시스템** DBMS, DataBase Management System 데이터베이스 구성과 연산을 통제하는 소프트웨어
- **데이터베이스 시스템** database system 데이터와 이를 저장할 저장 장치, 저장과 조회를 통제하는 소프트웨어 DBMS로 구성된 특정 시스템
- **데이터베이스 언어** database language 구조화된 데이터를 구성하고 수정하고 조회하는 언어
- **데커의 알고리즘** Dekker's Algorithm 두 스레드 간의 상호 배제를 보장하면서도 교착 상태나 무기한 연기를 방지해주는 알고리즘
- **덴트리(디렉토리 엔트리)** dentry(directory entry) **(리눅스)** 파일을 아이노드에 맵핑하는 구조
- **도메인** domain 관계형 데이터베이스 시스템에서 속성 값으로 가능한 값들의 집합
- **도메인 이름 시스템 공격** DNS(Domain Name System) attack 특정 웹사이트로 보내는 네트워크 트래픽의 주소를 수정하는 공격. 이러한 공격은 사용자를 특정 웹사이트에서 다른 사이트(해를 끼치는 사이트일 수 있다)로 돌리는 데 사용한다.
- **도출 속도** derived speed 프론트사이드 버스 속도와 클록 곱수나 제수를 통해 도출된 장치의 실제 속도
- **독립 소프트웨어 개발사** ISV, Independent Software Vendor 응용 소프트웨어를 개발해 판매하는 회사로, IBM PC 출시 이후 많은 소프트웨어 개발사가 생겨났다.
- **동기 신호** synchronous signal 현재 실행되고 스레드의 명령어 실행 결과로 발생하는 신호
- **동기화** synchronization 비동기 병행 스레드가 공유 자원에 순차적으로 접근하도록 조정하는 일
- **동시 발생** concurrent 이전 발생 관계보다 먼저 발생하는 쪽을 결정하기 어려운 두 이벤트를 '동시 발생' 이벤트라고 한다.
- **동적 RAM** DRAM, Dynamic RAM 내용을 메모리에 유지하기 위해 재생 회로 refresh circuit를 통해 계속해서 메모리 내용을 읽어오는 RAM
- **동적 로딩** dynamic loading 실행 시 메모리 주소를 지정하는 로딩 방법
- **동적 링킹** dynamic linking 프로세스가 맨 처음 호출할 때 외부 함수 참조를 확정하는 링킹 메커니즘. 동적 링킹은 프로세스가 실행되는 동안 전혀 참조하지 않는 외부 함수를 링크하지 않기 때문에 링킹으로 인한 오버헤드를 줄여준다.

- **동적 실시간 스케줄링 알고리즘** dynamic real-time scheduling algorithm 프로세스가 실행되는 동안 데드라인을 사용해 우선순위를 부여하는 스케줄링 알고리즘
- **동적 주소 변환** DAT, Dynamic Address Translation 실행 중에 가상 주소를 물리 주소로 변환하는 메커니즘. 실행 속도를 늦추지 않기 위해 아주 빠른 속도로 수행한다.
- **디렉토리** directory 다른 파일에 대한 참조를 저장하는 파일. 디렉토리 엔트리는 흔히 파일 이름과 유형, 크기 등을 포함한다.
- **디렉토리 엔트리 캐시** dcache(directory entry cache) **(리눅스)** 디렉토리 엔트리(즉, 덴트리)를 저장하는 캐시로, 커널이 빨리 파일 기술자를 대응하는 아이노드에 맵핑할 수 있게 해준다.
- **디바이스 드라이버** device driver 커널이 하드웨어 장치와 상호 작용할 수 있게 하는 소프트웨어. 디바이스 드라이버는 자신이 다루는 장치들과 매우 친밀하다. 예를 들면, 장치에 있는 데이터의 정렬 순서를 잘 알고 있고, 해당 장치에 맞게 데이터를 읽고 쓰며 DVD 드라이브의 트레이를 열고 닫는다. 드라이버는 시스템의 하드웨어가 바뀔 때, 더하거나 제거할 수 있도록 모듈 단위로 되어 있어서 시스템의 확장성을 더해주고, 이로써 사용자들은 새로운 장치를 쉽게 추가할 수 있다.
- **디스크 미러링** disk mirroring **(RAID)** RAID의 데이터 중복 기술로, 각 디스크 내용의 사본을 구별된 디스크에 저장한다. 이 기술은 높은 신뢰성을 제공하고 데이터 복구를 간략하게 해주지만, 상당한 저장소 오버헤드를 발생시키고 비용을 증가시킨다.
- **디스크 스케줄러** disk scheduler 운영체제 구성 요소로, 디스크 입출력 요청의 순서를 재정렬해 성능을 향상하는 역할을 한다.
- **디스크 스케줄링** disk scheduling 디스크 요청을 정렬해 처리량을 극대화하고 반응 시간과 그 변량을 최소화하는 기술. 디스크 스케줄링 전략은 탐색 시간과 회전 지연을 줄여 성능을 향상한다.
- **디스크 암** disk arm 무빙 헤드 디스크 구성 요소. 읽기/쓰기 헤드를 선형으로, 디스크 표면에 평행하게 움직인다.
- **디스크 암 예측** disk arm anticipation 다음 탐색을 최소화하는 위치로 디스크 암을 이동시키는 것. 디스크 암 예측은 디스크 요청 패턴이 지역성을 보이고 디스크 요청들 사이에 암을 이동시킬 충분한 시간이 있어서 성능을 저하시키지 않을 수 있는 시스템 환경에 유용하다.
- **디스크 재구성** disk reorganization 디스크의 파일 데이터를 옮겨 접근 시간을 개선하는 기술. 그 중 한 가지 기술은 조각 모음으로, 순차적인 파일 데이터를 연속된 디스크에 놓는다. 또 다른 기술은 자주 요청되는 데이터를 평균 탐색 시간이 적게 걸리는 트랙에 저장하기도 한다.
- **디스크 캐시 버퍼** disk cache buffer 운영체제가 디스크 데이터를 위해 예약해두는 메인 메모리 영역. 어떤 면에서 볼 때, 예약된 메모리는 캐시와 같은 작용을 해 프로세스들이 데이터에 빨리 접근할 수 있게 해준다. 예약된 메모리는 또한 버퍼와 같은 작용도 해 운영체제가 데이터 쓰는 것을 연기해 여러 입출력 연산을 적은 수의 요청으로 묶어 처리함으로써 입출력 성능을 향상한다.
- **디스패처** dispatcher 준비 리스트에 있는 첫 번째 프로세스를 프로세서에 할당해 실행할 수 있게 하는 운영체제 구성 요소

- **디스패칭** dispatching 프로세스에 프로세서를 할당하는 일
- **디지털 공증 서비스** digital notary service '타임스탬핑 대리자' 참고
- **디지털 서명** digital signature 수기 서명을 전자적으로 만든 서명. 디지털 서명을 생성하려면 송신자가 먼저 해시 함수를 원본 평문 메시지에 적용한다. 다음으로, 송신자는 자신의 비밀키를 사용해 메시지 다이제스트(즉, 해시 값)를 암호화한다. 이 단계에서는 디지털 서명을 생성하고 송신자의 신원을 검증한다. 해당 비밀키의 소유자만이 메시지를 암호화할 수 있기 때문이다.
- **디지털 서명 알고리즘** DSA, Digital Signature Algorithm 미국 정부의 디지털 인증 표준
- **디지털 엔벌로프** digital envelope 비밀키로 암호화한 메시지와 공개키 암호화를 사용해 암호화한 비밀키를 포함하는 패키지를 전송함으로써 메시지의 프라이버시를 보호하는 기술
- **디지털 워터마크** digital watermark 유명한 스테가노그라피 적용 예로, 파일 중 사용되지 않는(혹은 거의 사용되지 않는) 부분을 사용해 정보를 감춘다.
- **디지털 인증서** digital certificate 사용자나 기관을 식별하는 디지털 문서로 인증기관에서 발행한다. 디지털 인증서는 주체(기관이나 개인), 주체의 공개키, (인증서를 고유하게 식별하기 위한) 일련번호, 만료일, 신뢰성 있는 인증기관의 서명과 기타 관련 정보를 포함한다.
- **떠돌이 바이러스** transient virus 특정 컴퓨터 프로그램에 기생하는 바이러스. 이 바이러스는 프로그램을 실행하면 활성화되고, 프로그램을 종료할 때 비활성화된다.
- **라운드로빈 스케줄링** RR(Round-Robin) scheduling 준비 프로세스들이 한 라운드에 최소한 한 퀀텀을 소비하며 실행되도록 하는 스케줄링 정책. 큐에 있는 마지막 프로세스까지 실행을 마친 후, 스케줄러는 큐에 있는 첫 번째 프로세스로부터 새로운 라운드를 시작한다.
- **라이브러리 모듈** library module 미리 컴파일된 모듈로, 입출력이나 수학 함수 등 흔히 사용하는 루틴들을 수행한다.
- **랜덤 스캐닝 알고리즘** random-scanning algorithm 넓은 범위의 IP 주소를 생성하기 위해 의사 난수를 사용하는 알고리즘
- **램포트의 베이커리 알고리즘** Lamport's Bakery Algorithm 스레드 n개를 위한 상호 배제 알고리즘으로, '티켓 발급' 시스템에 기반하고 있다.
- **레인** lane PCI Express 버스에 있는 두 지점 간 통로. PCI Express 장치는 32개까지 레인으로 구성될 수 있는 링크로 연결된다.
- **레인달** Rijndael 벨기에의 조안 다먼과 빈센트 레이먼 박사가 개발한 블록 암호. 이 알고리즘은 다양한 프로세서에서 구현할 수 있다.
- **레지스터** register 프로세스에 위치한 고속 메모리로, 프로세스가 곧 사용할 데이터들을 저장한다.
- **레코드** record 데이터 계층에서 필드들의 그룹(예를 들면, 학생이나 고객에 관한 정보를 담은 연관된 몇몇 필드)
- **렉서** lexer '어휘 분석기' 참고

- **로그 구조 파일 시스템** LFS, Log-structured File System 모든 파일 연산을 트랜잭션으로 수행해 파일 시스템 데이터와 메타데이터가 항상 일치 상태에 있도록 보장하는 파일 시스템. LFS는 데이터가 시스템 전체의 로그 파일 뒤에 추가되므로, 일반적으로 좋은 쓰기 연산 성능을 보인다. 읽기 성능을 향상하기 위해 대체로 메타데이터를 로그 전체에 분산하고, 많은 캐시를 사용해 메타데이터를 저장해 파일 데이터의 위치를 빨리 찾아낼 수 있게 한다.
- **로그 파일** log file 운영 서비스가 요청된 시각과 이들을 요청하는 프로세스의 이름과 같은 시스템 동작에 대한 정보를 기록한다.
- **로더** loader 링크된 실행 가능 모듈을 메모리로 로드하는 응용 프로그램
- **로드 가능한 커널 모듈** loadable kernel module **(리눅스)** 실행 시에 커널에 통합되는 소프트웨어
- **로드 모듈** load module 링커가 만든 통합 모듈. 오브젝트 코드와 상대 주소로 구성된다.
- **로드 밸런싱** load balancing 시스템에 있는 프로세서 사이에 시스템 부하를 공평하게 분배하려는 연산
- **롤** roll '스왑' 참고
- **루시퍼 알고리즘** Lucifer algorithm IBM의 호르스트 파이스텔 Horst Feistel이 만든 암호화 알고리즘으로, 1970년대에 미국 정부와 미국 국가 안전 보장국 NSA, National Security Agency의 DES 알고리즘으로 채택되었다.
- **루트** root 파일 시스템의 구성의 시작점
- **루트 디렉토리** root directory 다양한 사용자 디렉토리를 가리키는 디렉토리
- **루트 사용자** root user **(리눅스)** '슈퍼유저' 참고
- **루트 키** root key 정책 생성 기관만을 위한 인증서 서명을 위해 인터넷 정책 등록 기관 IPRA, Internet Policy Registration Authority에서 사용한다.
- **루프백 장치** loopback device **(리눅스)** 시스템 서비스 계층 사이에서 데이터에 수행되는 연산을 가능하게 해주는 가상 장치
- **리눅스 보안 모듈 프레임워크** LSM(Linux Security Modules) framework **(리눅스)** 시스템 관리자가 시스템이 사용하는 접근 제어 메커니즘을 지정할 수 있게 해주는 프레임워크
- **리눅스 포트** port of Linux 다른 환경에서의 실행을 지원할 수 있도록 수정된 리눅스 커널 버전
- **리더/라이터 락** reader/writer lock **(리눅스)** 여러 스레드가 자원에서 읽기를 수행할 때 동시에 보유할 수 있고, 쓸 때는 한 스레드만 보유할 수 있는 락
- **리스트** list **(파일)** 파일의 내용을 보여주거나 출력하는 연산
- **릴레이션** relation 관계형 모델에서 튜플의 집합
- **링크** link 기존 파일을 참조하는 디렉토리 엔트리. 하드 링크는 파일이 위치한 저장 장치의 위치를 참조하고, 소프트 링크는 파일의 경로명을 저장한다.
- **링킹** linking 프로그램의 오브젝트 모듈을 실행 가능한 파일 하나로 통합하는 과정
- **링킹 로더** linking loader 링킹과 로딩을 모두 수행하는 응용 프로그램

- **마그네틱 테이프 저장소** magnetic tape storage 재기록 가능한 마그네틱 저장 매체로, 데이터에 순차적으로 접근한다. 이런 속성 때문에 직접 접근이 가능한 응용에서는 적합하지 못하다.
- **마더보드** motherboard '메인보드' 참고
- **마운트** mount 파일 시스템을 지역 디렉토리 구조에 삽입하는 일
- **마운트 연산** mount operation 이기종 파일 시스템을 한 네임스페이스로 결합하는 연산. 이러한 결합으로 파일들을 한 루트 디렉토리에서 접근할 수 있게 해준다.
- **마운트 지점** mount point 네이티브 파일 시스템에서 사용자가 지정한 디렉토리로, 마운트 명령어는 마운트된 파일 시스템의 루트를 해당 위치에 놓는다.
- **마운트 테이블** mount table 마운트 지점과 이에 대응하는 장치들의 위치를 저장하는 테이블
- **마이크로커널 운영체제** microkernel operating system 규모를 확장할 수 있는 운영체제로, 커널 안에 최소 서비스만 담고 있다. 다른 유형의 운영체제가 커널에 위임하는 많은 기능을 사용자 수준 프로그램에 요청한다.
- **마이크로코드** microcode 마이크로프로그래밍 명령어
- **마이크로프로그래밍** microprogramming 컴퓨터의 기계어보다도 저수준의 프로그래밍 계층으로, 기계어 연산을 구현하는 데 필요한 가장 기초적인 명령어를 포함한다. 이는 프로세서가 크고 복잡한 명령어들을 작게 나누어 프로세서의 실행 유닛이 수행할 수 있게 한다.
- **마크** Mach 초기 마이크로커널 운영체제로, 카네기멜론 대학교에서 리처드 라시드가 이끄는 팀에서 개발했다. 마크는 윈도우 NT 설계에 영향을 주었고, 맥 OS X을 구현하는 데 사용되었다.
- **만료 리스트** expired list **(리눅스)** 다음 시기까지는 프로세서를 위해 경쟁할 수 없는 프로세스들을 담고 있는 구조. 프로세스들은 이 리스트에 배치되어 다른 프로세스를 무기한 연기하는 일을 방지한다. 새로운 시기를 빨리 시작하기를 원할 때는 이 리스트를 활성 리스트로 만들면 된다.
- **만료 상태** expired state **(리눅스)** 작업이 다음 시기까지 디스패치되지 못하게 하는 작업 상태
- **매킨토시** Macintosh 애플 컴퓨터의 PC 계열로, 메인스트림 컴퓨터 사용자에게 GUI와 마우스를 선보였다.
- **맥 OS** Mac OS 1997년에 처음 도입된, 애플의 매킨토시 컴퓨터를 위한 운영체제 계열
- **멀티스레딩** multithreading 한 프로세스 내에 여러 실행 스레드를 두는 기술로, 동시에 병렬로 작동하는 일을 가능하게 해준다.
- **멀티프로그래밍** multiprogramming 여러 프로그램을 한꺼번에 메모리에 로드하고 동시에 실행하는 것
- **멀티프로그래밍 수준** level of multiprogramming '멀티프로그래밍 정도' 참고
- **멀티프로그래밍 정도** degree of multiprogramming 특정 시점에 메모리에 있는 프로세스의 총 수
- **멀틱스** Multics 초기 운영체제 중 하나로, 가상 메모리를 구현했다. MIT, GE, 벨 연구소 등에서 MIT의 CTSS 후속으로 개발했다.
- **멀틱스 MRDS** Multics Relational Data Store 최초의 상용 관계형 데이터베이스 시스템으로, 멀틱스에 포함되었다.

- **메모리 계층** memory hierarchy 메모리를 가장 빠르고 비싼 것에서부터 느리고 저렴한 메모리로 분류한 것. 일반적으로 상위 계층 메모리는 소용량, 하위 수준 메모리는 대용량으로 구성한다.
- **메모리 관리 장치** MMU, Memory Management Unit 특수 목적 하드웨어로, 가상-물리 주소 변환을 수행한다.
- **메모리 관리 전략** memory management strategy 특정 메모리 구성이 메모리 페치와 배치, 교체 등을 어떻게 수행할 것인지 명시하는 전략
- **메모리 관리자** memory manager 시스템의 메모리 구성과 메모리 관리 전략을 구현한 운영체제 구성 요소
- **메모리 구성** memory organization 시스템이 메인 메모리를 보는 방식으로, 메모리에 프로세스가 몇 개까지 존재하는지, 프로그램과 데이터를 메모리의 어디에 둘 것인지, 언제 그 부분들을 다른 부분들과 교체할 것인지 등을 고려한다.
- **메모리 덤프** memory dump **(리눅스)** 프로세스를 종료하기 전에 코어 파일을 생성하는 것
- **메모리 맵 파일** memory-mapped file 데이터가 프로세스의 가상 주소 공간에 맵핑되는 파일. 이는 프로세스가 파일 데이터를 다른 데이터에 접근하는 것처럼 참조할 수 있게 해준다. 메모리 맵 파일은 파일 데이터에 빈번하게 접근하는 프로그램에 유용하다.
- **메모리 보호** memory protection 프로세스들이 다른 프로세스나 운영체제에서 사용하는 메모리에 함부로 접근하는 일을 방지하는 메커니즘
- **메모리 사용량** memory footprint **(리눅스)** 커널이 소비한 스왑핑할 수 없는 메모리 크기
- **메모리 압축** memory compaction 가변 파티션 멀티프로그래밍 시스템에서 여기저기 산재한 모든 파티션을 메인 메모리의 한쪽 끝에 재배치해 가능하면 큰 여유 메모리 영역을 생성하는 일
- **메모리 트림시키기** burping the memory '메모리 압축' 참고
- **메모리 풀** memory pool **(리눅스)** 커널이 메모리를 별도로 예약해 어떤 프로세스가 향후 메모리 요청을 해올 때 거부하지 않도록 보장하는 메모리 영역
- **메모리 홀 병합** coalescing memory holes 가변 파티션 멀티프로그래밍 시스템에서 메모리의 인접 홀들을 병합하는 과정. 이를 통해 들어오는 프로그램과 데이터가 사용할 수 있는 가장 큰 여유 공간을 만들 수 있다.
- **메소드** method **(객체)** 객체의 속성들을 조작하고, 서비스를 수행하는 부분
- **메시지** message 메시지 유형과 가변 길이 필드를 지정해 데이터를 전송할 수 있게 하는 IPC 메커니즘
- **메시지 다이제스트** message digest SHA-1, MD5와 같은 알고리즘을 메시지에 적용해 생성한 해시 값
- **메시지 무결성** message integrity 메시지 전송 중에 공격자가 변경하지 않았음을 나타내는 속성
- **메시지 전달** message passing 관련이 없는 프로세스들이 데이터를 교환해 통신할 수 있게 해주는 메커니즘
- **메시지 큐** message queue **(리눅스)** 프로세스에 아직 전달되지 않은 메시지들을 저장하는 자료 구조
- **메시지 큐 기술자** message queue descriptor **(리눅스)** 메시지 큐에 관한 데이터를 저장하는 자료 구조
- **메인 메모리** main memory 명령어와 데이터를 저장하는 휘발성 메모리. 메모리 계층에서 프로세서가 직접 참조할

수 있는 최하위 수준 메모리

- **메인보드** mainboard 프로세스나 메모리, 주변 장치 등 컴퓨터 구성 요소 간 전기적 연결을 제공하는 인쇄 회로 기판
- **메타데이터** metadata 파일 시스템이 파일을 관리하는 데 사용하는 데이터. 메타데이터는 사용자들이 직접 접근할 수 없다. 아이노드와 슈퍼블록은 메타데이터의 예다.
- **멤버** member 파티션된 파일에서 순차적인 하위 파일
- **명령어 길이** instruction length 주어진 특정한 아키텍처에서 한 명령어를 구성하는 비트 수. 어떤 아키텍처는 다양한 길이의 명령어를 지원하기도 한다. 명령어 길이는 아키텍처에 따라 변한다.
- **명령어 페치기** instruction fetch unit 프로세서 구성 요소로, 명령어 캐시에서 명령어를 로드해 명령어를 해석하고 실행하게 한다.
- **명령어 해석기** instruction decode unit 프로세서 구성 요소로, 명령어들을 해석하고 적절한 제어 신호를 생성해 프로세스가 각 명령어를 실행하게 한다.
- **명시적 승인** explicit acknowledgement 클라이언트가 서버로부터 응답을 받을 때, 클라이언트가 추가적인 패킷을 사용해 서버에 승인 acknowledge 신호를 보내는 일
- **모놀리식 운영체제** monolithic operating system 커널 안에 운영체제 구성 요소를 모두 포함한 운영체제로, 대개 커널은 컴퓨터 시스템에 제한 없이 접근할 수 있다.
- **모니터** monitor 병행성 관련 구조로, 순차적으로 재사용 가능한 공유 자원이나 이러한 자원의 그룹을 할당하기 위한 상호 배제 메커니즘을 제공하는 데 필요한 데이터와 프로시저를 모두 가지고 있다.
- **모니터 진입 루틴** monitor entry routine 어떤 스레드든 호출할 수 있는 모니터 루틴. 그러나 한 번에 한 스레드에서만 실행할 수 있다. 모니터 진입 루틴은 모니터 내부에서 실행하는 스레드에서만 호출할 수 있는 private 모니터 루틴과 달리 상호 배제를 강제할 수 있다.
- **모듈** module 독립적으로 개발되는 하위 프로그램으로, 다른 하위 프로그램들과 결합되어 더 크고 복잡한 프로그램을 형성한다. 프로그래머들은 미리 컴파일된 라이브러리 모듈을 사용해 입출력 조작이나 난수 발생 등 자주 사용하는 컴퓨터 기능을 수행한다.
- **무결성** integrity **(보안 트랜잭션)** 성공적이고 안전한 트랜잭션을 위한 다섯 가지 기본 요소 중 하나로, 주고받는 데이터가 손상되거나 변경되지 않았음을 보장하는 문제를 다룬다.
- **무기한 연기** indefinite postponement 스레드가 결코 발생하지 않을 수도 있는 이벤트를 대기하는 상황
- **무빙 암 어셈블리** moving-arm assembly '디스크 암' 참고
- **무어의 법칙** Moore's law 프로세서 설계의 발전에 관한 예측으로, 프로세서의 트랜지스터 수가 약 18개월마다 배가된다고 예상했다.
- **문맥 교환** context switching 운영체제가 프로세서에서 어느 한 프로세스를 제거하고 다른 프로세스를 실행할 때 수행되는 일로, 운영체제는 교체되는 프로세스의 상태 정보를 저장해둬야 한다. 또한 프로세서에 다시 할당될 때 이 정보를 사용해 프로세스를 원래 상태로 복구해야 한다.

- **문자** character 데이터 계층 구조에서 고정 길이 비트 패턴으로, 일반적으로 8비트, 16비트, 32비트로 되어 있다.
- **문자 장치** character device 키보드나 마우스와 같은 장치로, 한 번에 한 바이트 데이터를 전송한다. 고정 크기 바이트 그룹을 전송하는 블록 장치와 대비된다.
- **문자 집합** character set 문자들의 집합. 많이 알려진 문자 집합으로는 아스키 ASCII, EBCDIC, 유니코드 Unicode 등이 있다.
- **문제 상태** problem state '사용자 모드' 참고
- **물리 레코드** physical record 디스크에서 실제로 읽거나 쓰는 정보 단위
- **물리 메모리** physical memory '메인 메모리' 참고
- **물리 뷰** physical view 데이터가 저장된 특정 장치와 데이터 포맷, 데이터가 전송되는 물리적 수단 등과 관련해 파일 데이터를 보는 관점
- **물리 블록** physical block '물리 레코드' 참고
- **물리 장치 이름** physical device name 파일 이름을 특정 장치에 맞게 지정한 이름
- **물리 주소** physical address '실제 주소' 참고
- **물리 주소 공간** physical address space 특정 컴퓨터에서 메인 메모리의 크기에 대응하는 물리적 주소 범위. 물리 주소 공간은 흔히 각 프로세스의 가상 주소 공간보다 작다.
- **물리 주소 확장** PAE, Physical Address Extension **(IA-32 인텔 아키텍처)** IA-32 프로세서가 64GB까지의 메인 메모리 주소를 관리할 수 있게 하는 메커니즘
- **물리적 백업** physical backup 저장 장치의 각 비트를 복사하는 일. 파일 시스템의 내용을 해석하려는 노력은 전혀 없다.
- **미국 표준 기술 연구소** NIST, National Institute of Standards and Technology 암호화나 다른 표준들을 지정하는 미 정부 기관
- **미들웨어** middleware 서로 다른 응용 프로그램이 통신할 수 있게 하는 소프트웨어 계층. 네트워크 통신이나 포맷이 다른 데이터를 해석해주는 등 응용 프로그래밍을 단순화해준다.
- **미러링** mirroring **(RAID)** '디스크 미러링' 참고
- **미세 단위 스트립** fine-grained strip **(RAID)** 평균적인 파일들이 여러 스트라이프에 저장되게 하는 스트립 크기. 미세 단위 스트립은 각 요청의 접근 시간을 줄이고 전송 시간을 증가시킨다. 여러 디스크에서 요청된 데이터의 부분들을 동시에 조회하기 때문이다.
- **미션 크리티컬 시스템** mission-critical system 반드시 올바로 수행해야 하는 시스템으로, 시스템이 실패할 경우 금전적 손실뿐 아니라 인명 피해까지 초래할 수 있다.
- **미지 상태** unknown state **(윈도우 XP)** 어떤 오류가 발생해 시스템이 알지 못하는 스레드 상태
- **바쁜 대기** busy waiting 스레드가 다음 단계로 진행하기 위해 계속해서 조건을 확인하면서 대기하는 것. 바쁜 대기

동안에 스레드는 프로세서 시간을 소비한다.

- **바운스 버퍼** bounce buffer **(리눅스)** 메모리 영역 중 커널이 상위 메모리 지역 high memory zone의 데이터를 직접 참조할 수 있는 곳에 맵핑하도록 하는 부분. 바운스 버퍼는 시스템의 물리 주소 공간이 커널의 가상 주소 공간보다 클 때 반드시 필요하다.
- **바이너리 버디 알고리즘** binary buddy algorithm **(리눅스)** 리눅스가 물리 페이지 프레임을 할당하는 데 사용하는 알고리즘으로, 연속적인 페이지 그룹의 리스트를 유지한다. 각 그룹의 수는 2의 거듭제곱이다. 이 알고리즘은 연속적인 물리 메모리에 접근해야 하는 프로세스나 장치에 메모리를 할당하는 일을 수월하게 해준다.
- **바이너리 세마포어** binary semaphore 값이 1을 넘지 않는 세마포어로, 대개 자원 하나를 할당하는 데 사용한다.
- **바이러스** virus 실행 가능한 코드로 흔히 이메일 메시지에 첨부되거나 오디오 클립, 비디오 클립, 게임 등의 파일에 숨겨 전송되며, 다른 파일에 첨부되거나 덮어쓰기해서 자신을 복제한다. 대체로 자신이 거주하는 시스템에 해를 입힌다.
- **바이러스 서명** virus signature 바이러스 생성 사이에 변경되지 않는 코드 세그먼트
- **바이오스** BIOS, Basic Input/Output System 기본적인 하드웨어 초기화와 관리를 제어하는 저수준 소프트웨어 명령어들
- **바이트** byte 데이터 계층 구조에서 두 번째로 낮은 수준. 바이트는 대체로 8비트로 구성된다.
- **바이트코드** bytecode 가상 머신에서 작동하는 중간 코드. 예를 들어, 자바 바이트코드는 자바 가상 머신에서 작동한다.
- **방화벽** firewall 외부 네트워크의 악의적인 사용자가 보낸 패킷으로부터 LAN을 보호하는 소프트웨어나 하드웨어
- **배치 전략** placement strategy **(메인 메모리)** 들어오는 프로그램과 데이터를 메인 메모리의 어디에 둘지 결정하는 전략
- **배치 프로세스** batch process 사용자와 상호 작용 없이 실행되는 프로세스
- **배타적 논리합 연산** XOR(eXclusive-OR) operation 두 비트가 동일하면 1, 다르면 0을 반환하는 연산. RAID 수준 3~5는 XOR 연산을 사용해 패리티 비트를 생성한다.
- **배포판** distribution **(리눅스)** 리눅스 커널과 사용자 응용 프로그램 그리고 설치 과정을 간략하게 해주는 도구를 담고 있는 소프트웨어 패키지
- **백도어 프로그램** back-door program 상주 바이러스로, 공격자가 들키지 않고 피해자의 컴퓨터 자원에 접근해 작업을 완료할 수 있게 해준다.
- **백업** backup 정보의 사본을 생성하는 일
- **버스** bus 메인보드에 있는 다른 장치 사이의 정보 교환을 위한 고속 통신 채널로 트레이스의 집합으로 이루어진다.
- **버스 마스터링** bus mastering 장치가 메모리에 접근하기 위해 버스를 제어함을 가정한 DMA 전송(다른 장치들이 버스에 동시에 접근하는 일을 막아줌)

- **버퍼** buffer 다른 속도로 작동하는 장치들이 입출력하는 동안 데이터를 임시로 저장하는 공간. 버퍼는 속도가 빠른 장치가 자기 속도를 발휘해 버퍼의 용량만큼 데이터를 생산하게 하면서 그동안 느린 장치가 데이터를 소비하도록 기다린다.
- **버퍼 오버플로우** buffer overflow 할당된 공간보다 큰 입력 값을 보내는 공격. 입력을 적절히 코딩하고 시스템의 스택이 실행 가능하면 버퍼 오버플로우는 공격자가 악의적인 코드를 실행할 수 있게 만든다.
- **범용 레지스터** general-purpose register 프로세스들이 데이터와 포인터 값을 저장하는 데 사용하는 레지스터. 특수 목적 레지스터는 사용자 프로세스에서 접근할 수 없다.
- **베라디의 이상 현상** Belady's Anomaly 'FIFO 이상 현상' 참고
- **베어울프 클러스터** Beowulf cluster **(리눅스)** 고성능의 병렬 처리 시스템으로, 컴퓨터 클러스터들로 구성되며, 각각은 리눅스 커널에 대한 베어울프 수정판을 실행한다.
- **변경 불가 속성** immutable attribute **(리눅스)** 파일을 읽고 실행할 수 있지만 복사, 수정, 삭제할 수 없음을 나타내는 파일 속성
- **변경 불가 파일** immutable file 생성 후에는 수정할 수 없는 파일
- **변위** displacement 블록이나 페이지 혹은 세그먼트에서 특정 주소까지의 거리로, 오프셋이라고도 한다.
- **변종** variant 원래의 형태에서 수정된 바이러스로, 여전히 악성 페이로드에 남아 있다.
- **변환 참조 버퍼** TLB, Translation Lookaside Buffer 고속 연관 메모리 맵으로, 가상 페이지 수와 대응하는 페이지 프레임 수 사이의 소수의 맵핑을 보유한다. TLB는 대체로 최근 사용된 페이지 테이블 엔트리를 저장해 지역성을 보이는 프로세스들의 경우 성능을 향상해준다.
- **병렬 포트** parallel port 프린터와 같은 병렬 입출력 장치의 인터페이스
- **병목 현상** bottleneck 자원이 처리할 수 있는 속도보다 빨리 요청들이 쇄도해 프로세스 실행이 느려지고 자원 활용도가 떨어지는 상태. 하드 디스크들은 대부분의 시스템에서 병목 현상을 일으킨다.
- **병행** concurrent 프로세스나 스레드가 다른 프로세스나 스레드와 동시에 존재하는 상태
- **병행 프로그램 실행** concurrent program execution 프로세서 시간이 여러 활성 프로세스 사이에서 공유되게 하는 기술. 단일 프로세서 시스템에서는 병행 프로세스를 병행으로 실행할 수 없다. 멀티프로세서 시스템에서만 병행 실행이 가능하다.
- **보류 신호** pending signal 스레드에 전달되지 않은 신호. 스레드가 실행되지 않고 있거나, 스레드가 해당 유형의 신호를 마스킹한 경우에 발생한다.
- **보안 메커니즘** security mechanism 시스템이 자체 보안 정책을 구현하는 방법. 많은 시스템에서 정책은 시간이 지남에 따라 변하지만, 보안 메커니즘은 변하지 않은 채 유지된다.
- **보안 모델** security model 시스템의 주체, 객체, 권한을 정의하는 개체
- **보안 소켓 계층** SSL, Secure Sockets Layer 넷스케이프 커뮤니케이션즈에서 개발한 비독점 프로토콜로, 인터넷에 있는 두 컴퓨터 사이의 통신에서 보안성을 갖추게 한다.

- **보안 정책** security policy 시스템 자원에 대한 접근을 관장하는 규칙
- **보안 패치** security patch 보안 허점을 해결하는 코드
- **보안 해시 알고리즘** SHA-1, Secure Hash Algorithm NIST에서 개발한 유명한 해시 함수로 디지털 서명을 생성하는 데 사용한다.
- **보안성 있는 운영체제** secure operating system 사용자와 소프트웨어가 허가받지 않은 서비스나 데이터에 접근하는 일을 방지해주는 운영체제
- **보조 저장소** auxiliary storage '2차 저장소' 참고
- **보조 저장소 관리** auxiliary storage management 파일 시스템의 구성 요소로, 2차 저장 장치에 파일을 위한 공간을 할당하는 일을 담당한다.
- **보조 프로세서** coprocessor 그래픽 프로세서나 디지털 신호 프로세서 등 제한된 몇 가지 명령어를 효율적으로 수행하기 위해 고안된 특수 목적 프로세서
- **보호** protection 시스템의 보안 정책을 구현하는 메커니즘으로, 응용 프로그램이 허가받지 않은 자원이나 서비스에 접근하지 못하게 방지한다.
- **보호 도메인** protection domain 접근 권한의 모음. 보호 도메인 내의 각 접근 권한은 객체의 이름과 적용할 수 있는 권한 필드를 갖는 순서 쌍으로 표현한다.
- **보호 변수** protected variable **(세마포어)** 세마포어의 상태를 저장하는 정수 값으로, 해당 세마포어에 P 혹은 V 연산을 호출하는 방법으로만 접근하거나 수정할 수 있다.
- **복구** recovery 시스템 다운 후 데이터를 회복하는 일
- **복사** copy **(파일)** 새 이름을 가진 다른 버전의 파일을 생성하는 연산
- **복제** replication 한 시스템에서 동일한 기능을 수행하는 여러 자원을 제공하는 것
- **복제 투명성** replication transparency 시스템 사용 가능한 여러 시스템 자원이 있다는 사실을 감추는 것. 자원에 대한 모든 접근은 마치 그 자원 하나만 있는 것처럼 보인다.
- **복호화** decryption 데이터 암호화를 역으로 수행해 원본 형태를 읽을 수 있게 하는 기술
- **볼륨** volume 여러 파일을 보유할 수 있는 저장 단위
- **부 버전 번호** minor version number **(리눅스)** 리눅스 커널의 연속적인 안정 버전(짝수)과 개발 버전(홀수)을 식별하는 값
- **부 장치 식별 번호** minor device identification number **(리눅스)** 동일한 메이저 번호를 할당받은 장치들을 고유하게 식별하기 위한 값
- **부모 디렉토리** parent directory 계층 구조 파일 시스템에서 현재 디렉토리의 한 단계 상위 디렉토리
- **부모 프로세스** parent process 자식 프로세스를 하나 이상 생성한 프로세스. 유닉스 시스템에서는 fork 시스템 호출을 사용해 자식 프로세스를 생성한다.

- **부분 순서** partial ordering 이전 발생 관계'를 따르는 이벤트 순서. 어떤 이벤트들은 이 체계를 통해 순서를 정할 수 없다. 그렇기 때문에 부분 순서가 된다.
- **부인 방지** nonrepudiation 메시지 전송 혹은 수신 사실을 증명하는 방법
- **부재 세그먼트 폴트** missing-segment fault 프로세스가 현재 메인 메모리에 없는 세그먼트를 참조할 때 발생하는 오류. 이에 대한 반응으로 운영체제는 2차 저장소에서 해당 세그먼트를 로드한다.
- **부트 섹터** boot sector 초기 운영체제 명령어가 저장되는 디스크의 특정 위치. 바이오스는 컴퓨터에 전원이 들어올 때 하드웨어가 이 명령어들을 로드하도록 한다.
- **부트 섹터 바이러스** boot sector virus 컴퓨터 하드 디스크의 부트 섹터를 감염해 운영체제와 함께 로드되어 시스템의 제어권을 장악하는 바이러스
- **부트스트랩핑** bootstrapping 운영체제 구성 요소의 초기 부분을 시스템의 메모리에 로드해 운영체제의 나머지 부분을 로드할 수 있게 하는 과정
- **부하** load '요청률' 참고
- **분산 교착 상태** distributed deadlock 단일 프로세서 시스템에서의 교착 상태와 유사한 것이지만, 프로세스들이 다른 컴퓨터 사이에 퍼져 있는 환경에서 발생하는 교착 상태
- **분산 교착 상태 탐지 전략** distributed deadlock detection strategy 분산 시스템에서 교착 상태를 찾아내는 데 사용하는 기술
- **분산 데이터베이스** distributed database 네트워크에 있는 컴퓨터 시스템 전체에 걸쳐 분산된 데이터베이스
- **분산 서비스 거부 공격** DDoS(Distributed Denial-of-Service) attack 시스템이 요청을 적절히 서비스하지 못하게 하는 공격. 개별 컴퓨터에서 일제히 패킷을 보내게 해 트래픽을 발생시키는 방법을 사용한다.
- **분산 시스템** distributed system 공통 작업을 수행하기 위해 네트워크를 통해 협력하는 원격 컴퓨터의 집합
- **분산 운영체제** distributed operating system 전통적인 운영체제와 같은 서비스를 제공하지만, 시스템 내의 객체들은 해당 서비스를 제공하는 컴퓨터를 알지 못하도록 적절한 투명성을 제공해야 한다.
- **분산 컴퓨팅** distributed computing 공동 작업을 위해 독립적인 컴퓨터 여러 대를 사용하는 방식
- **분산 파일 시스템** distributed file system 네트워크에 있는 컴퓨터 시스템 전체에 걸쳐 분산된 파일 시스템
- **분할할 수 없는 연산** indivisible operation '원자적' 연산 참고
- **불릿 서버** bullet server 표준 파일 서버로, 아메바 Amoeba 파일 시스템에서 사용한다.
- **불안전 상태** unsafe state 다익스트라의 은행원 알고리즘에서 나오는 시스템 상태로, 시스템이 보유한 자원이 프로세스들에 충분하지 않아서 교착 상태가 발생할 가능성이 있는 상태다.
- **불연속 메모리 할당** noncontiguous memory allocation 메모리 할당 방법으로, 한 프로그램을 여러 부분으로 나누어 인접되지 않은 메모리 영역에 배치하는 일
- **붐** boom '디스크 암' 참고

- **브루트포스 크랙킹** brute-force cracking 단순하게 가능한 모든 패스워드를 시도해 알아내거나, 가능한 모든 복호화 키를 사용해 메시지를 복호화해서 시스템의 보안을 뚫는 기술
- **블록** block 고정 크기의 데이터 단위로, 보통 한 바이트보다 훨씬 크다. 연속적인 데이터 레코드를 블록에 놓으면 시스템이 이들을 조회할 때 입출력 연산의 수를 줄일 수 있다.
- **블록 그룹** block group **(리눅스)** 연속적인 블록들의 모음으로, 그룹 전체에 걸친 자료 구조로 관리한다. 따라서 관련 데이터 블록, 아이노드, 기타 파일 시스템 메타데이터들이 디스크에 연속적으로 놓인다.
- **블록 리스트** blocked list 모든 블록된 프로세스를 가리키는 포인터를 담은 커널 자료 구조. 이 리스트는 특별한 우선순위로 유지되지는 않는다.
- **블록 맵 테이블** block map table 프로세스의 가상 블록을 대응하는 메인 메모리에 맵핑하는 엔트리들을 담은 테이블. 가상 메모리 시스템에서 블록은 세그먼트 혹은 페이지다.
- **블록 맵 테이블 시작점 레지스터** block map table origin register 프로세스의 블록 맵 테이블의 메인 메모리 주소를 저장하는 레지스터. 고속 레지스터는 가장 주소 변환을 빨리 수행할 수 있도록 돕는다.
- **블록 맵핑** block mapping 가상 메모리 시스템에서 가상 메모리 주소와 실제 메모리 주소 사이의 맵핑 수를 줄이는 방법으로, 개별적인 주소 대신 블록 단위로 맵핑한다.
- **블록 상태** blocked state 프로세스(또는 스레드)가 특정 이벤트의 완료(예를 들면, 입출력 완료)를 기다리며 대기하는 상태로, 블록 상태 프로세스(또는 스레드)는 프로세서가 쉬고 있어도 사용할 수 없다.
- **블록 암호** block cipher 메시지를 고정 크기 비트 그룹으로 나누어 암호화 알고리즘을 적용하는 암호화 기술
- **블록 장치** block device 디스크와 같이 고정 크기 바이트 그룹으로 데이터를 전송하는 장치로, 한 번에 한 바이트 데이터를 전송하는 문자 장치와 대비된다.
- **블록 할당** block allocation 파일 시스템이 2차 저장소를 더 효율적으로 관리하도록 해주는 기술. 파일에 영역 extent(연속적인 섹터 블록)을 할당함으로써 파일 탐색 오버헤드를 줄여준다.
- **블록 할당 비트맵** block allocation bitmap **(리눅스)** 각 블록 그룹의 블록 사용을 추적하는 비트맵
- **블록 해제** unblock 대기하던 이벤트가 완료된 후 블록 상태 프로세스를 제거하는 일
- **블록킹** blocking 연속된 레코드를 큰 블록들로 그룹화하는 것. 이 블록들은 한 번의 입출력 연산을 통해 읽을 수 있다. 이 기술은 많은 레코드들을 한 번의 입출력 연산으로 조회함으로써 접근 시간을 감소시킬 수 있다.
- **블록화되지 않은 레코드** unblocked record 각 물리적 레코드에 대해 정확히 논리적 레코드 하나를 갖는 레코드
- **블록화된 레코드** blocked record 각 물리 레코드마다 논리적인 레코드 몇 개를 담을 수 있는 레코드
- **비균일 요청 분산** nonuniform request distribution 디스크 표면 전체에 균일하게 분포되지 않는 디스크 요청들. 이는 프로세스들이 공간적 지역성을 보여 요청 패턴이 특정 지역에 집중되기 때문에 발생한다.
- **비동기 병행 스레드** asynchronous concurrent thread 동시에 존재하지만 서로 독립적으로 동작하며, 이따금씩 통신하고 동기화해 협력적인 작업을 수행하는 스레드들

- **비동기 신호** asynchronous signal 실행 중인 스레드의 명령어와 상관없이 발생하는 신호
- **비동기 실시간 프로세스** asynchronous real-time process 이벤트에 대한 반응으로 실행되는 실시간 프로세스
- **비동기 전송** asynchronous transmission 블록킹의 필요를 줄이려고 한 장치에서 버퍼를 통해 독립된 다른 장치로 데이터를 전송하는 일. 일단 데이터가 버퍼에 도착하면 송신자는 수신자가 아직 데이터를 읽지 않은 상태에서도 다른 일을 수행할 수 있다.
- **비동기 취소** asynchronous cancellation **(POSIX)** 취소 신호를 받자마자 스레드가 종료되는 취소 모드
- **비밀키 암호 기법** secret-key cryptography 동일한 비밀키를 가지고 암호화와 복호화를 수행하는 기술. 송신자는 비밀키를 사용해 메시지를 암호화하고, 암호화된 메시지를 원하는 수신자에게 보낸다. 수신자는 동일한 비밀키를 가지고 메시지를 복호화한다. 이 기법은 '대칭형 암호화 기법'이라고도 한다.
- **비선점형 스케줄링** nonpreemptive scheduling 프로세스가 작업을 완료하거나 자발적으로 반납할 때까지는 시스템이 프로세서를 빼앗지 않도록 하는 스케줄링 정책
- **비율 단조 스케줄링** RM(Rate-Monotonic) scheduling 프로세스가 디스패치되어야 하는 비율에 비례하는 우선순위를 할당하는 실시간 스케줄링 정책
- **비즈니스 크리티컬 시스템** business-critical system 적절하게 기능하지 못할 경우, 기업의 생산성이나 수익성을 떨어뜨릴 수 있는 시스템. 실패할 경우 인명 피해 등 막대한 해를 입히는 미션 크리티컬 시스템과 비교하면 심각성이 덜 하다.
- **비트 패턴** bit pattern 데이터 계층 구조의 가장 낮은 수준. 비트 패턴은 사실상 컴퓨터 시스템에서 관심을 두는 모든 데이터 항목을 나타내는 비트 그룹이다.
- **비트맵** bitmap 여유 공간을 관리하는 기술로, 메모리의 각 블록마다 한 비트를 유지하며, i번째 비트는 메모리의 i번째 블록에 대응한다. 비트맵은 파일 시스템이 더 쉽게 연속적인 블록을 할당할 수 있게 해주지만, 여유 블록을 찾는 데 상당한 실행 시간을 소비하게 된다.
- **비활성 리스트** inactive list **(리눅스)** '만료 리스트' 참고
- **비활성 페이지** inactive page **(리눅스)** 메인 메모리의 페이지 중 들어오는 페이지로 교체할 수 있는 페이지
- **비활성화 프로세스** deactivated process **(리눅스)** 실행 큐에서 제거되어 더는 프로세서 시간을 얻으려고 경쟁할 수 없는 프로세스
- **비활성화된 취소** disabled cancellation **(POSIX)** 스레드가 보류 중인 취소 신호를 받지 않는 취소 모드
- **빅 커널 락** BKL, Big Kernel Lock **(리눅스)** 전역 스핀 락으로 리눅스 커널에서 SMP를 지원한 초기의 구현이다.
- **빠른 상호 배제 알고리즘** fast mutual exclusion algorithm 다른 스레드가 임계 영역에 들어가려고 경쟁하지 않는 경우에, 여러 번의 테스트 수행에 드는 오버헤드를 피하는 상호 배제 구현 방법. 최초의 빠른 상호 배제 알고리즘은 램포트가 제안했다.
- **사용성 있는 운영체제** usable operating system 사용하기 쉬운 인터페이스와 사용자 중심 응용 프로그램을 통해 여러 사용자 계층을 서비스할 수 있는 운영체제

- **사용자 디렉토리** user directory 각 사용자의 파일에 해당하는 엔트리를 담고 있는 디렉토리. 각 엔트리는 해당 파일이 저장 장치에 저장된 위치를 가리킨다.
- **사용자 모드** user mode 프로세스가 시스템 자원에 직접 접근하는 일을 금지하는 동작 모드
- **사용자 모드 리눅스** UML, User-Mode Linux **(리눅스)** 호스트 리눅스 시스템 안에서 사용자 프로세스처럼 실행되는 리눅스 커널
- **사용자 상태** user state '사용자 모드' 참고
- **사용자 수준 스레드** user-level thread 프로세스 내의 모든 스레드가 한 실행 문맥에 할당되는 스레딩 모델
- **사용자 클래스** user class 개별 사용자나 사용자 그룹을 파일에 접근할 수 있는 권한에 따라 분류하는 스키마
- **사이클** cycle **(클록)** 전기 신호의 한 진폭. 초당 사이클 수는 장치(예를 들면, 프로세서나 메모리, 버스 등)의 주파수를 결정하며, 시스템이 시간을 측정하는 단위로 사용할 수 있다.
- **사이클 가로채기** cycle stealing 버스에 접근할 때 프로세서보다 채널에 우선권을 주어, 채널에서 온 신호와 프로세서가 충돌하는 일을 방지하는 방법
- **삭제** delete **(파일)** 파일에서 데이터 항목을 삭제하는 연산
- **산술 논리 장치** ALU, Arithmetic and Logic Unit 프로세서의 구성 요소로, 기본적인 산술 연산과 로직 연산을 수행한다.
- **삼중 DES** Triple DES DES의 변종으로, 블록에 각각 다른 비밀키를 적용하는 세 DES 시스템 시리즈로 간주할 수 있다. '3DES'라고도 한다.
- **삼중 간접 포인터** triply indirect pointer 이중 간접 포인터 블록의 위치를 가리키는 아이노드 포인터
- **삽입** insert **(파일)** 파일에 새로운 데이터를 추가하는 연산
- **상대 경로** relative path 파일의 위치를 현재 작업 디렉토리와의 관계에 따라 지정하는 경로
- **상대 주소** relative address 모듈의 시작 지점으로부터의 위치를 기반으로 지정한 주소
- **상위 메모리** high memory **(리눅스)** 물리 메모리의 영역으로 (IA-32 아키텍처의 경우에는 896MB에서 시작한다), 영구적으로 커널의 가상 주소 공간에 맵핑된 가장 큰 물리 주소에서 시작해, 물리 메모리의 제한선(인텔 펜티엄 4 프로세서의 경우 64GB에 해당한다)까지 확장된다. 커널은 페이지를 가상 주소 공간에서 고수준 메모리의 페이지 프레임으로 맵핑하는 값비싼 연산을 수행해야 하므로, 대부분의 커널 자료 구조는 고수준 메모리에 저장된다.
- **상주 바이러스** resident virus 메모리에 한번 로드되면 컴퓨터의 전원이 나갈 때까지 활동하는 바이러스
- **상주 페이지 집합** resident page set 현재 메모리에 존재하는 페이지들의 집합. 이 페이지들은 페이지 폴트를 발생시키지 않고 참조할 수 있다. 상주 페이지 집합은 프로세스의 작업 집합의 크기와 다를 수 있다. 작업 집합은 프로세스가 효율적으로 실행되기 위해 메모리에 있어야 좋은 페이지들의 집합이다.
- **상태 전이** state transition 프로세스의 상태가 어느 한 상태에서 다른 상태로 변하는 것
- **상태 정보** state information 자원의 상태를 기술하는 데이터

- **상한 레지스터** limit register 고정 파티션 멀티프로그래밍 시스템에서 한 프로세스의 메모리 파티션이 끝나는 지점을 표시하는 레지스터
- **상호 배제** mutual exclusion 어느 한 스레드가 자신의 임계 영역을 수행할 때 다른 스레드들이 자신의 임계 영역에 들어가는 것을 막는 제한. 상호 배제는 여러 스레드가 수정 가능한 공유 데이터를 접근하는 프로그램이 정확하게 동작하는 데 꼭 필요한 메커니즘이다.
- **상호 배제 잠금** mutual exclusion lock 어느 한 스레드가 자신의 임계 영역에서 수행하고 있는지 여부를 나타내는 변수. 만일 잠금이 어느 한 스레드가 자신의 임계 영역에 있음을 나타내면 다른 스레드들은 자신의 임계 영역 외부에 잠금된다.
- **상호 배제 프리미티브** mutual exclusion primitive 상호 배제를 구현하는 데 필요한 가장 기본적인 연산으로, enterMutualExclusion()과 exitMutualExclusion()을 예로 들 수 있다.
- **상호 운용성** interoperability 소프트웨어 컴포넌트가 이기종 하드웨어와 소프트웨어 플랫폼, 프로그래밍 언어, 통신 프로토콜 사이에서 상호 작용할 수 있게 해주는 속성
- **생산자** producer 데이터를 생산해 공유 객체에 두는 프로세스나 스레드
- **생산자 스레드** producer thread 데이터를 생산해 공유 객체에 두는 스레드
- **생산자/소비자 관계** producer/consumer relationship 데이터를 생산하는 스레드와 데이터를 소비하는 스레드 사이의 상호 작용. 이 모델은 비동기식 병행 실행에서 발생하는 여러 미묘한 문제를 보여준다.
- **생성** create **(파일)** 새 파일을 만드는 연산
- **생체 인식** biometrics 사용자를 식별하기 위해 지문이나 안구 홍채 스캔, 안면 스캔 등 개인의 신체적 특징을 사용하는 기술
- **섀도우 패스워드 파일** shadow password file **(유닉스)** 암호화된 패스워드 이외의 정보를 일반 패스워드 파일에 저장하고 암호화된 패스워드는 루트 권한이 있는 사용자만 접근할 수 있는 섀도우 패스워드 파일에 저장해 크랙커들로부터 패스워드 파일을 보호한다.
- **섀도우 페이지** shadow page 수정된 내용을 새로운 블록에 쓰는 데이터 블록. 섀도우 페이지는 트랜잭션을 구현하는 한 가지 방법이다.
- **섀도우 페이징** shadow paging 수정된 블록을 새 블록에 쓰는 트랜잭션 구현. 수정되지 않은 블록의 사본은 트랜잭션이 커밋될 때 여유 공간으로 해제된다.
- **섀도잉** shadowing **(RAID)** '미러링' 참고
- **서명 스캐닝 바이러스 탐지** signature-scanning virus detection 바이러스 코드에 대한 지식에 의존하는 안티바이러스 기술
- **서버** server 클라이언트라는 다른 프로세스에 서비스를 제공하는 프로세스. 이러한 프로세스들을 실행하는 컴퓨터 역시 서버라고 한다.
- **서버 스텁** server stub RPC에서 서버 측 스텁으로, 외부로 나가는 데이터를 전송할 수 있도록 준비시키고, 들어오

는 데이터를 변환해 올바르게 해석할 수 있게 한다.

- **서비스 거부 공격** DoS(Denial-of-Service) attack 시스템이 합법적인 요청을 적절히 서비스하지 못하도록 하는 공격. 많은 서비스 거부 공격에서 인가되지 않은 트래픽들이 네트워크 자원을 포화시켜 합법적인 사용자가 접근하는 것을 방해한다. 대체로 이 공격은 데이터 패킷으로 서버에 포화를 일으키는 방법으로 이루어진다.
- **서비스 티켓** service ticket **(커베로스)** 클라이언트가 특정 네트워크 서비스에 접근하는 것을 허가해주는 티켓
- **선입선출** FIFO, First-In-First-Out 비선점형 스케줄링 방식으로, 준비 큐에 도착하는 순서에 따라 프로세스들을 디스패치한다.
- **선입선출 페이지 교체** FIFO(First-In-First-Out) page replacement 메모리에 가장 오래 머문 페이지를 교체 대상으로 선택하는 페이지 교체 전략. FIFO 방식은 오버헤드는 적지만 미래의 페이지 사용을 정확히 예측하기는 어렵다.
- **선점 가능 자원** preemptible resource 프로세서나 메모리와 같이 프로세스들로부터 빼앗을 수 있는 자원. 이러한 유형의 자원은 교착 상태에 연루되지 않는다.
- **선점 락 카운터** preemption lock counter **(리눅스)** 커널 모드에서 실행되는 커널이 선점될 수 있는지 여부를 결정하기 위해 사용되는 정수 값. 이 카운터 값은 선점될 수 없는 동안에 커널 제어 경로가 임계 영역을 마주칠 때마다 증가한다.
- **선점 불가능한 자원** nonpreemptible resource 프로세스들로부터 강제로 빼앗을 수 없는 자원(예를 들면, 테이프 드라이브). 이러한 유형의 자원은 교착 상태에 연루될 수 있다.
- **선점형 스케줄링** preemptive scheduling 시스템이 프로세스로부터 프로세서를 빼앗을 수 있는 스케줄링 정책
- **선착 우선 서비스 디스크 스케줄링** FCFS(First-Come-First-Served) disk scheduling 가장 먼저 도착한 요청이 먼저 서비스되는 디스크 스케줄링 전략. FCFS는 공평한 방법이지만 요청률(즉, 부하)이 커질 때 대기 시간이 길어진다. FCFS는 가장 안쪽 실린더에서 가장 바깥쪽 실린더에 이르기까지 임의의 탐색 패턴을 보이며, 연속된 요청들도 많은 탐색 시간이 들 수 있다.
- **선형 순서** linear ordering **(하벤더)** 자원들을 논리적으로 정렬한 것으로, 프로세스들은 선형 순서로 자원을 요청해야 한다. 이 방법은 순환 대기 조건을 거부해 교착 상태를 방지한다.
- **선형 주소 공간** linear address space **(IA-32 인텔 아키텍처)** 32비트(4GB) 가상 주소 공간. 순수 세그먼테이션 방식에서 이 주소 공간은 메인 메모리에 직접 맵핑된다. 세그먼테이션/페이징 조합 방식에서는 메인 메모리에 맵핑되는 페이지 프레임으로 나뉜다.
- **선호 방향** preferred direction SCAN 기반 스케줄링 알고리즘에서 디스크 헤드가 움직이는 방향
- **설정 시간** setup time 시스템 운영자와 운영체제가 다음 작업을 실행 준비시키기는 데 필요한 시간
- **세그먼트** segment 프로세스의 가상 주소 공간에서 가변 크기의 연속적 주소 집합. 세그먼트는 대체로 (프로시저의 명령어 집합이나 배열의 내용 등과 같은) 유사 항목 전체의 크기일 때가 많다. 이는 시스템이 그러한 항목을 적절한 접근 권한을 사용해 섬세하게 보호할 수 있게 해준다. 예를 들어, 데이터 세그먼트는 대체로 실행 권한이 아닌, 읽기 전용이나 읽기/쓰기 권한을 부여받는다. 이와 비슷하게 실행 가능한 명령어들을 담고 있는 세그먼트에

는 쓰기 권한이 아니라 보통, 읽기/실행 권한이 주어진다. 세그먼트들은 외부 단편화를 초래하는 경향이 있지만, 내부 단편화는 없다.

- **세그먼트 기술자** segment descriptor **(IA-32 인텔 아키텍처)** 세그먼트 맵 테이블 개체로, 세그먼트의 기준 주소, 존재 비트, 경계 주소, 보호 비트를 저장한다.
- **세그먼트 맵 테이블 시작점 레지스터** segment map table origin register 프로세스의 세그먼트 맵 테이블의 메인 메모리 위치를 담고 있는 레지스터. 이 정보를 고속 레지스터에 유지함으로써 가상-물리 주소 변환을 빠르게 수행할 수 있다.
- **세그먼트 보호 예외** segment-protection exception 프로세스가 접근 제어 모드에서 명시한 것과 다른 방식으로 접근할 때 발생하는 예외
- **세그먼트 선택기** segment selector **(IA-32 인텔 아키텍처)** 대응하는 세그먼트 기술자(즉, 세그먼트 맵 테이블 엔트리)가 있는 세그먼트 맵 테이블의 변위(오프셋)를 가리키는 16비트 값
- **세그먼트 오버플로우 예외** segment-overflow exception 세그먼트 밖의 주소를 참조하려고 할 때 발생하는 예외
- **세마포어** semaphore 두 원자적 연산(P와 V)을 사용해 보호되는 정수 변수(스레드가 자신의 임계 영역에 들어갈 수 있는지를 결정하는 변수)에 접근하게 하는 상호 배제 추상화 개념
- **세마포어 배열** semaphore array **(리눅스)** 연관된 자원들에 대한 접근을 보호하는 세마포어의 연결 리스트
- **세션키** session key (고객이 온라인 상점에서 물건을 구매하는 등의) 트랜잭션 기간 동안 사용하는 비밀키
- **섹터** sector 입출력 요청에 따라 접근할 수 있는 트랙의 가장 작은 부분
- **섹터 큐잉** sector queuing '회전 최적화' 참고
- **셸** shell 사용자가 GUI나 텍스트를 통해 운영체제와 상호 작용할 수 있게 해주는 응용 프로그램
- **소비자** consumer 공유 객체로부터 데이터를 읽어가서 처리하는 응용 프로그램
- **소비자 스레드** consumer thread 공유 객체로부터 데이터를 읽어가서 처리하는 것을 목적으로 하는 스레드
- **소스 코드** source code 고급 언어나 어셈블리 언어로 작성한 프로그램 코드. 이런 소스 코드를 컴퓨터가 이해할 수 있으려면 먼저 컴파일하거나 해석해야 한다.
- **소스 트리** source tree **(리눅스)** 소스 코드 파일과 디렉토리를 담은 구조. 모놀리식 리눅스 커널에 대한 논리적인 구성을 제공한다.
- **소유자** owner **(파일 접근 제어)** 파일을 생성한 사용자
- **소켓** socket 프로세스들이 직접적인 통신 채널을 수립해 데이터를 교환할 수 있게 해주는 IPC 메커니즘. 프로세스들이 read, write 호출을 사용해 네트워크를 통해 통신할 수 있게 해준다.
- **소켓 쌍** socket pair **(리눅스)** 서로 연결된 이름 없는 소켓의 쌍. 단일 시스템의 프로세스 간에 양방향 통신에 사용할 수 있다.
- **소켓 주소** socket address 소켓의 고유한 식별자

- **소프트 링크** soft link 링크되는 파일에 대응하는 경로명을 지정하는 파일
- **소프트 실시간 스케줄링** soft real-time scheduling 실시간 프로세스에 그렇지 않은 프로세스들보다 높은 우선순위를 부여함을 보장하는 스케줄링 정책
- **소프트웨어 인터럽트 처리기** software interrupt handler **(리눅스)** 인터럽트 처리 코드로, 인터럽트를 마스킹하지 않고 수행될 수 있으기 때문에 선점될 수 있다.
- **속성** attribute **(객체)** '프로퍼티' 참고
- **손상** wound 프로세스가 다른 프로세스에 손상을 입을 때, 해당 프로세스는 롤백된다.
- **손상 대기 교착 상태 방지 전략** wound-wait deadlock prevention strategy 비선점 조건을 거부함으로써 교착 상태를 방지하는 전략. 생성 시기를 기준으로 각 프로세스에 고유한 속성을 부여한다. 다른 프로세스가 점유한 자원을 요청하는 프로세스는 만일 해당 프로세스보다 먼저 생성된 경우, 해당 프로세스에 손상을 줄 것이다. 만약 대기하는 대상 프로세스보다 나중에 생성되었다면 대기할 것이다.
- **솔라리스** Solaris 시스템 V 릴리즈 4와 SunOS 모두에 기반을 둔 유닉스 버전. AT&T와 썬이 협력해 설계했다.
- **수정 비트** modified bit 페이지가 수정되어 교체 전에 2차 저장소로 복사되어야 하는지 여부를 나타내는 페이지 테이블 엔트리 비트. 더티 dirty bit라고도 한다.
- **수준** level **(RAID)** 수준 1(미러링), 수준 2(해밍 ECC 패리티) 등 RAID 시스템의 특정 구현 방식을 나타낸다.
- **순수 페이징** pure paging (세그먼테이션 없이) 페이징 방식만 사용하는 메모리 구성 기술
- **순차적 파일 구성** sequential file organization 레코드들이 물리적으로 순차적인 순서로 정렬되는 파일 구성 기술. '다음' 레코드는 물리적으로 이전 레코드를 뒤따르는 레코드다.
- **순차적으로 재사용 가능한 공유 자원** serially reusable shared resource 한 번에 한 스레드에서만 사용할 수 있는 자원
- **순차적으로 재사용 가능한 자원** serially reusable resource '전용 자원' 참고
- **순차적으로 재사용 가능한 코드** serially reusable code 수정 가능하지만 사용될 때마다 초기화되는 코드. 이러한 코드는 한 번에 한 프로세스나 스레드에서만 사용할 수 있다.
- **순차화** serialize 공유 변수에 대한 접근을 통제해, 한 번에 한 스레드만 접근할 수 있게 하는 일. 다른 스레드는 처음에 들어간 스레드가 임계 영역에서 작업을 완료하고 나온 후에야 진입할 수 있다.
- **순환 대기** circular wait 교착 상태 성립 조건으로, 프로세스 둘 이상이 '순환 고리'로 갇혀 있으면서 각 프로세스가 다음 프로세스가 보유하고 있는 자원을 얻으려고 대기하는 상황
- **슈퍼바이저 상태** supervisor state '커널 모드' 참고
- **슈퍼바이저 호출** supervisor call 사용자 프로세스가 운영체제에 자신을 대신해서 연산을 수행해 달라고 요청하는 일(시스템 호출이라고도 함)
- **슈퍼블록** superblock 파일 시스템의 무결성에 중요한 정보(예를 들면, 파일 시스템의 여유 블록 리스트나 비트맵의 위치, 파일 시스템 식별자, 파일 시스템 루트 위치 등)를 담는 블록

- **슈퍼유저(루트 사용자)** superuser(root user) **(리눅스)** (커널이나 시스템에 해를 입힐 수도 있는) 제한된 연산을 수행할 수 있는 사용자
- **스래싱** thrashing 페이징 활동이 과도하게 발생해 프로세스 활용도가 떨어지는 것. 프로세스의 메모리 할당이 작업 집합에 비해 적을 경우 발생한다. 이 경우, 프로세스가 페이지들이 2차 저장소로부터 메모리에 로드되는 것을 기다리느라 상당한 시간을 소비하게 해 성능이 저하된다.
- **스레드** thread 독립적으로 실행할 수 있는 프로그래밍 명령어 집합을 설명하는 개체(실행 흐름, 혹은 제어 흐름이라고도 한다). 스레드는 한 프로세스 안에서 병행 활동의 병렬 실행을 가능하게 해준다.
- **스레드 상태** thread state 실행, 준비, 블록 등의 스레드 상태
- **스레드 지역 저장소** TLS, Thread Local Storage 윈도우 XP에서 스레드에 특정한 데이터를 구현하는 것으로, 스레드는 자신의 주소 공간을 프로세스 내의 다른 스레드들과 공유할 수 있지만, 자신만 접근할 데이터는 TLS에 저장한다.
- **스레드 취소** cancellation of a thread 대상 스레드를 종료하는 스레드 연산. 스레드 취소에는 취소 비활성화, 취소 지연, 비동기적 취소라는 세 가지 취소 모드가 있다.
- **스레드 풀링** thread pooling 몇몇 커널 스레드가 자신을 생성한 프로세스가 실행되는 동안 계속 유지되게 하는 기술로, 스레드 풀링은 비용이 많이 드는 스레드 생성과 종료 연산을 줄여 성능을 높여준다.
- **스마트 카드** smart card 신용 카드 크기의 데이터 저장소로, 인증과 데이터 저장소 기능을 한다.
- **스왑** swap 프로세스의 메모리 내용을 2차 저장소로 복사해, 프로세스를 메모리에서 제거하고 해제된 메모리를 새 프로세스에 할당하는 일
- **스왑 명령어** swap instruction 두 변수의 값을 원자적으로 교환하는 연산. 이 명령어는 스레드가 '읽기-수정-쓰기' 연산을 수행하는 도중에 선점당할 가능성을 제거함으로써 상호 배제 구현을 간략하게 해준다.
- **스왑 캐시** swap cache **(리눅스)** 페이지 테이블 엔트리들의 캐시. 특정 페이지가 2차 저장소의 시스템 스왑 파일에 있는지 여부를 나타낸다. 만일 페이지 테이블 엔트리가 스왑 캐시에 있으면 대응하는 페이지가 스왑 파일에 존재하며, 스왑 파일에 쓸 필요가 없게 된다.
- **스캐너** scanner '어휘 분석기' 참고
- **스케줄러 활성화** scheduler activation 사용자 수준 라이브러리가 커널 스레드를 스케줄링할 수 있게 하는 기술
- **스켈레톤** skeleton 서버 측 스텁
- **스택 영역** stack region 프로세스의 주소 공간 중 호출된 프로시저를 위해 명령어들과 값들을 저장하는 영역. 스택의 내용은 프로세스가 중첩되게 프로시저들을 호출할 때마다 늘어나고, 호출된 프로시저들이 작업을 마치고 반환할 때마다 줄어든다.
- **스텁** stub 외부로 나가는 데이터를 전송할 수 있도록 준비시키고, 들어오는 데이터를 변환해 올바르게 해석할 수 있게 하는 것
- **스테가노그라피** steganography 다른 정보 안에 정보를 숨기는 기술. 이 용어는 '숨겨진 글'이라는 의미의 라틴어 어

원에서 나왔다.

- **스트라이프** stripe **(RAID)** RAID 시스템에서 각 디스크의 동일한 위치에 있는 스트립들의 집합. 스트라이핑은 RAID 시스템이 한 번에 여러 디스크를 사용해 파일들에 접근할 수 있게 해주며, 이는 전송 시간을 개선시켜 준다.
- **스트림 소켓** stream socket TCP 프로토콜을 사용해 데이터를 전송하는 소켓
- **스트립** strip **(RAID)** RAID 시스템에서 데이터가 조작되는 최소 단위. 각 디스크에서 동일한 위치에 있는 스트립들의 집합을 스트라이프라고 한다.
- **스풀** spool, Simultaneous Peripheral Operations OnLine 프로세스가 2차 저장소에 데이터를 기록해서 속도가 느린 장치들로 전송될 때까지 버퍼링될 수 있게 하는 입출력 방법
- **스핀 락** spin lock 임계 영역에 대한 상호 배타적인 접근을 제공해주는 락. 프로세스가 잠금을 보유한 채 자신의 임계 영역 안에서 실행되고 있을 때, 다른 프로세서에서 동시에 실행되는 프로세스가 자신의 임계 영역에 들어가기 전에 잠금을 얻으려고 시도하면 바쁜 대기 busy wait 상태에 들어가게 된다.
- **스핀들** spindle 무빙 헤드 디스크 구성 요소로 빠른 속도로 플래터들을 회전시킨다.
- **슬랩** slab **(리눅스)** 작은 구조 때문에 생기는 내부 단편화를 줄여주는 메모리 페이지. 한 페이지 크기보다 작은 여러 구조를 저장한다.
- **슬랩 캐시** slab cache **(리눅스)** 최근 사용된 슬랩들을 저장하는 캐시
- **슬랩 할당자** slab allocator **(리눅스)** 슬랩 캐시에 놓이는 객체들을 위해 메모리를 할당해주는 커널 개체
- **시간 클록** time-of-day clock 컴퓨터 시스템 외부에서 인식하는 실제 시간을 측정하는 시계로, 수천~수백만 분의 1초 오차 정도의 정확성이 있다.
- **시간적 지역성** temporal locality 이벤트들이 시간에 걸쳐 밀접하게 연관되는 속성. 메모리 참조 시, 프로세스들이 짧은 기간 안에 동일한 메모리 위치를 계속해서 참조할 때 시간적 지역성 현상이 발생한다고 말한다.
- **시공간 지수** space-time product 프로세스의 실행 시간의 product(즉, 프로세스가 메모리를 차지하고 있는 기간)를 측정하는 값. 이상적으로는 메모리 관리를 통해 이 값을 줄여서 시스템의 멀티프로그래밍 정도를 높여야 한다.
- **시기** epoch **(리눅스)** 모든 프로세스가 스케줄러의 활성 리스트에서 만료 리스트로 이동하는 시간. 시기 개념을 도입함으로써 프로세스들이 무기한 연기되는 일을 방지한다.
- **시분할 시스템** timesharing system 동시에 여러 사용자가 시스템과 상호 작용할 수 있게 지원하는 운영체제
- **시분할 시스템** TSS, Time Sharing System 1960년대에 IBM에서 설계한 운영체제로, 시분할과 가상 메모리 기능을 제공했다. 상용 제품으로 출시되지는 않았지만 훗날 이 시스템의 많은 기능 특성이 IBM 시스템에 나타났다.
- **시스템 침투** system penetration 허가되지 않은 외부 사용자가 컴퓨터 보안을 성공적으로 뚫은 것
- **시스템 파일 시스템** sysfs, system file system **(리눅스)** 프로세스들이 통합 장치 모델을 통해 정의한 구조에 접근할 수 있게 해주는 파일 시스템

- **시스템 프로그래밍** systems programming 시스템의 장치들과 응용 프로그램을 관리하는 소프트웨어를 개발하는 일
- **시스템 호출** system call 운영체제의 서비스를 요청하는 프로시저 호출. 프로세스가 시스템 호출을 발생할 때 프로세서 실행 모드가 사용자 모드에서 커널 모드로 바뀌어 운영체제가 해당 호출에 대한 응답으로 적절한 명령어를 실행한다.
- **시한폭탄** time bomb 컴퓨터의 시계가 특정 시간이나 날짜가 되면 활성화되는 바이러스
- **신뢰성** reliability 장애 내구성 척도. 더 신뢰성 있는 자원일수록 고장 날 확률이 낮아진다.
- **신호** signal 특정 이벤트나 오류가 발생한 사실을 알려주려고 소프트웨어에서 보낸 메시지. 신호는 수신자에게 데이터를 보낼 수는 없다.
- **신호** signal **(세마포어)** 세마포어의 변수 값을 증가시키는 연산. 만일 해당 세마포어를 대기하며 휴면하는 스레드가 있으면, 신호 연산은 그 중 하나를 깨우고 세마포어 값을 1 감소시킨다.
- **신호 마스크** signal mask 어떤 신호가 스레드에 전달되지 않도록 지정하는 자료 구조. 신호 유형과 기본 동작에 따라, 마스크된 신호는 큐에 쌓이거나 손실될 수 있다.
- **신호 마스킹** mask a signal 신호가 전달되는 것을 막는 방법. 신호 마스킹은 멀티스레드 프로세스가 자신의 스레드 중 어떤 것이 특정 유형의 신호를 처리할 것인지 지정할 수 있게 해준다.
- **신호 처리기** signal handler 특정 유형의 신호에 반응해 실행되는 코드
- **신호 처리기의 기본 동작** default action for a signal handler **(리눅스)** 미리 지정된 신호 처리기로, 프로세스가 대응하는 신호 처리기를 지정하지 않았을 경우, 신호에 반응해 실행되는 동작
- **신호 후 계속 모니터** signal-and-continue monitor 스레드가 모니터 사용 가능함을 신호할 수 있게 하면서도, 해당 스레드가 모니터를 나갈 때까지는 잠금을 해제하지 않을 수 있게 하는 모니터. 이때 신호를 받은 스레드가 모니터에 들어갈 수 있다.
- **신호 후 종료 모니터** signal-and-exit monitor 스레드가 다른 스레드에 신호를 보내는 즉시 모니터에 대한 잠금을 해제하게 하는 모니터
- **실린더** cylinder 디스크 암의 특정 위치에 있는 읽기/쓰기 헤드를 통해 접근할 수 있는 트랙들의 집합
- **실시간 스캐너** real-time scanner 메모리에 상주하면서 활발하게 바이러스를 예방하는 소프트웨어
- **실시간 스케줄링** real-time scheduling 시간적 제약에 근거해 우선순위를 결정하는 스케줄링 정책
- **실시간 시스템** real-time system 일정 시간(대체로 매우 짧은 시간) 안에 서비스를 수행해야 하는 시스템으로, 미션 크리티컬 실시간 시스템인 경우 제시간에 실행 결과를 내지 못하면 재정적 손실은 물론 인명 피해까지 초래할 수 있다.
- **실시간 신호** real-time signal **(리눅스)** 유실되는 신호가 없도록 보장해 실시간 시스템을 구현할 수 있도록 도와주는 신호 구현
- **실제 메모리** real memory '메인 메모리' 참고
- **실제 시간** wall clock time '벽시계 시간', 즉 사용자가 인식하는 시간 척도

- **실제 주소** real address 메인 메모리에 있는 주소
- **실행 모드** execution mode 운영체제의 실행 모드(사용자 모드 혹은 커널 모드 등)로, 프로세스에서 어떤 명령어를 실행할 수 있는지를 결정한다.
- **실행 상태** running state 프로세스(스레드)가 프로세서에서 실행하고 있는 상태
- **실행 접근** execute access **(가상 메모리)** 프로세스가 페이지나 세그먼트에서 명령어를 실행할 수 있는 접근 권한
- **실행 접근** execute access **(파일)** 사용자가 파일을 실행할 수 있게 허용하는 권한
- **실행 큐** run queue **(리눅스)** 특정 프로세서에서 실행하려고 대기하는 프로세스들의 리스트
- **심볼 테이블** symbol table 오브젝트 모듈의 부분으로, 한 모듈에 있는 각 외부 명칭과 외부 참조 목록을 보여준다.
- **심볼 확정** symbol resolution 링커를 통해 한 모듈 안에 있는 외부 참조를 다른 모듈에 있는 외부 명칭과 매칭하는 과정
- **심볼릭 이름** symbolic name 장치 독립적인 이름(예를 들어, 경로명)
- **싱글 사인 온** single sign-on 사용자가 한 번 로그인하면 동일한 패스워드를 사용할 수 있도록 해 인증 과정을 간략화한다.
- **쓰기** write **(파일)** 프로세스의 출력 데이터를 파일에 쓰는 연산
- **쓰기 시 복사** copy-on-write 프로세스 생성 효율을 높이는 메커니즘으로, 프로세스가 페이지를 수정할 때까지 부모와 자식 간에 맵핑 정보를 공유한다. 수정하는 시점에는 해당 페이지의 새로운 사본이 생성되어 해당 프로세스에 할당된다. 이는 부모나 자식 프로세스가 공유 페이지 중 많은 부분을 수정할 때는 상당한 오버헤드를 야기할 수 있다.
- **쓰기 접근** write access **(가상 메모리)** 프로세스가 페이지나 세그먼트에 내용을 쓸 수 있도록 허용하는 접근 권한
- **쓰기 접근** write access **(파일)** 파일에 쓸 수 있도록 허용하는 권한
- **아스키** ASCII, American Standard Code for Information Interchange 개인용 컴퓨터와 데이터 통신 시스템에서 많이 사용하는 문자 집합으로, 문자들을 8비트 바이트로 저장한다.
- **아이노드** inode 유닉스 기반 시스템에서 파일 제어 블록과 단일, 이중, 삼중 간접 파일 포인터들의 블록을 담은 인덱스 블록
- **아이노드** inode **(리눅스)** 파일, 디렉토리 또는 파일 시스템의 링크에 대응하는 데이터 블록의 위치를 나타내는 구조체. VFS에서는 이 구조가 시스템의 파일을 나타낸다. ext2 아이노드는 ext2 파일 시스템의 파일을 나타낸다.
- **아이노드 맵** inode map 로그 구조 파일 시스템의 로그에 사용하는 메타데이터 블록으로, 파일 시스템의 아이노드 위치를 나타낸다. 아이노드 맵은 LFS에서 파일의 위치를 결정하는 데 걸리는 시간을 줄여 LFS 성능을 향상한다.
- **아이노드 캐시** inode cache **(리눅스)** 아이노드 검색 성능을 향상하는 캐시
- **아이노드 테이블** inode table **(리눅스)** 블록 그룹에서 할당된 각 아이노드에 대한 엔트리를 담고 있는 구조체
- **아이노드 할당 비트맵** inode allocation bitmap **(리눅스)** 블록 그룹의 아이노드 사용을 기록하는 비트맵

- **아키텍처 특정 코드** architecture-specific code 특정 아키텍처에 맞게 고유한 명령어들을 지정하는 코드
- **안전 상태** safe state 다익스트라의 은행원 알고리즘에서의 시스템 상태. 시스템에 있는 모든 프로세스가 교착 상태에 빠지지 않고 모두 작업을 마칠 수 있는 동작 순서가 존재하는 상태다.
- **안티바이러스 소프트웨어** antivirus software 바이러스를 식별하고 제거하는 등 시스템을 바이러스로부터 보호하는 소프트웨어
- **암시적 승인** implicit acknowledgement 요청 메시지에 대한 승인을 암시하는 응답 메시지
- **암호** cipher 메시지를 암호화하는 수학적 알고리즘으로, '암호 체계'라고도 한다.
- **암호 기법** cryptography 데이터 암호화와 복호화 기술을 사용해 의도된 수신자만 데이터를 해석할 수 있게 하는 기법
- **암호 분석 공격** cryptanalytic attack 복호화키를 소유하지 않은 사람이 암호문을 해독하려고 시도하는 기술. 가장 흔한 공격은 암호화 알고리즘을 분석해 암호화키와 암호문과의 관계를 알아내는 것이다.
- **암호 체계** cryptosystem 메시지를 암호화하는 수학적 알고리즘으로, '암호'라고도 한다.
- **암호문** ciphertext 암호화된 데이터
- **암호화** encryption 데이터를 변형해 인가되지 않는 사용자가 해석하지 못하게 하는 기술
- **암호화 API** Cryptographic API **(리눅스)** 응용 프로그램과 서비스들이 데이터를 암호화 하고 복호화할 수 있는 커널 인터페이스
- **암호화 파일 시스템** EFS, Encrypting File System 암호 기법을 사용하는 NTFS 특징으로, 윈도우 XP 프로페셔널과 윈도우 2000에서 파일과 폴더를 보호한다. EFS는 비밀키와 공개키 암호 기법을 사용해 파일을 보호한다. 각 사용자는 키 쌍을 할당받고 해당 사용자만이 파일을 암호화했고 이에 접근할 수 있음을 보증하는 인증서를 할당받는다.
- **애드온 카드** add-on card 컴퓨터의 기능을 확장하는 장치로, 사운드 카드나 비디오 카드를 들 수 있다.
- **액츄에이터** actuator '디스크 암' 참조
- **어셈블러** assembler 어셈블리 언어 프로그램을 기계어로 변환하는 프로그램
- **어셈블리 언어** assembly language 기본적인 컴퓨터 기능을 영어와 유사한 약어로 표현한 저수준 프로그래밍 언어
- **어휘 분석기** lexical analyzer 컴파일러의 일부로, 소스 코드를 토큰 단위로 분리한다.
- **에어 갭 기술** air gap technology 방화벽을 보완하는 네트워크 보안 해결책. 내부 네트워크에 접근하는 외부 사용자로부터 사적인 데이터의 보안을 유지한다.
- **에이다** Ada 1970년대와 1980년대에 미 국방부에서 개발한 절차적, 병행 프로그래밍 언어
- **에이징** aging 프로세스가 자원을 대기하는 시간에 따라 우선순위를 높여주어 무기한 연기를 방지하는 방법
- **엔트리 집합** entry set 자바에서 동기화 메소드를 호출한 후, 모니터에 들어가려고 대기하고 있는 스레드들의 큐
- **엔트리 큐** entry queue '엔트리 집합' 참고

- **엘리베이터 알고리즘** elevator algorithm 'SCAN 디스크 스케줄링' 참고
- **여유 리스트** free list 여유 블록의 주소를 담고 있는 연결 리스트 블록
- **여유 메모리 리스트** free memory list 운영체제 자료 구조로, 메모리의 여유 공간을 가리킨다.
- **여유 시간** laxity 데드라인으로부터 현재 시간과 데드라인까지 남은 실행 시간의 합을 빼서 계산해낸 값. 이 값은 프로세스가 데드라인에 가까워질수록 줄어든다.
- **역 맵핑** reverse mapping **(리눅스)** 메모리의 페이지를 참조하는 페이지 테이블 엔트리들의 연결 리스트. 역 맵핑은 이제 곧 교체되려는 공유 페이지에 대응하는 모든 PTE를 갱신하는 일을 도와준다.
- **역 페이지 테이블** inverted page table 메인 메모리에 있는 페이지 프레임마다 한 엔트리를 포함하는 페이지 테이블. 역 페이지 테이블은 페이지 프레임보다 훨씬 많은 수의 페이지 테이블 엔트리를 메모리에 유지하는 전통적인 페이지 테이블보다 단편화를 적게 발생한다. 해시 함수는 가상 페이지 번호를 역 페이지 테이블의 인덱스에 맵핑한다.
- **역할** role **(RBAC)** 조직의 구성원에 부여한 작업 집합. 각 역할은 각 역할의 사용자가 접근할 수 있는 객체를 정의하는 일련의 특권(접근 권한)을 부여받는다.
- **역할 기반 접근 제어** RBAC, Role-Based Access Control 사용자에게 역할을 부여하는 접근 제어 모델
- **연관 맵핑** associative mapping 내용 기반 주소 연관 메모리로 가상 주소를 대응하는 실제 메모리 주소로 맵핑하는 것을 돕는다. 연관 메모리의 모든 개체를 동시에 검색한다.
- **연관 메모리** associative memory 위치가 아닌 내용으로 검색하는 메모리. 빠른 연관 메모리는 고속 동적 주소 변환 메커니즘을 돕는다.
- **연속 메모리 할당** contiguous memory allocation 프로세스 전체의 주소 영역이 모두 서로 인접하도록 메모리를 할당하는 방법
- **열기** open **(파일)** 파일을 참조할 수 있게 준비하는 연산
- **영구 저장소** persistent storage '2차 저장소' 참고
- **영속 투명성** persistence transparency 자원이 저장된 위치(메모리든 디스크든)를 감추는 것
- **영역** extent 연속적인 섹터들의 블록
- **예외** exception 프로세스 때문에 발생한 오류. 프로세서 예외는 운영체제를 호출해 어떻게 반응할지 결정하게 한다. 프로세스는 예외 처리기를 등록해 운영체제가 해당 유형의 예외를 받을 때 해당 처리기를 실행하게 할 수 있다.
- **예외** exception **(인텔 IA-32 명세)** 오류 때문에 생성되는 하드웨어 신호. 인텔 IA-32 명세에서는 예외를 트랩, 폴트, 중단으로 분류한다.
- **예측 가능성** predictability 시스템 부하와 상관없이 프로세스가 일정한 시간이 걸리도록 보장하는 스케줄링 알고리즘 목표

- **예측 버퍼링** anticipatory buffering 메인 메모리에서 한 번에 레코드 하나 이상을 버퍼링함으로써 프로세싱과 입출력 연산이 오버랩되게 하는 기술
- **예측 이동** anticipatory movement 디스크 암 예측 방식에서의 디스크 암의 움직임
- **예측 페이징** anticipatory paging 프로세스가 가까운 미래에 사용할 확률이 높은 페이지를 메모리에 미래 로드하는 것. 이러한 전략을 프로세스가 페이지 폴트를 덜 만나게 하려는 노력의 일환이다.
- **예측 페치 전략** anticipatory fetch strategy 페이지가 요청되기 전에 미리 메인 메모리에 로드해, 요청될 때 즉시 사용 가능할 수 있게 준비시키는 방법. 이는 프로그램이 다음에 수행할 연산을 예측함으로써 수행된다.
- **오렌지북** Orange Book 미 국방부에서 펴낸 문서로, 운영체제의 보안 기능을 평가하는 가이드라인을 제시한다.
- **오브젝트 코드** object code 컴파일러가 생성한 코드로, 기계어 명령어를 담고 있다. 오브젝트 코드는 먼저 링크와 로딩을 수행한 후에 실행할 수 있다.
- **오프셋** offset '변위' 참고
- **오픈 소스 소프트웨어** OOS, Open-Source Software 소스 코드를 포함하며, GPL 혹은 이와 유사한 라이선스로 배포되는 소프트웨어. 오프 소스 소프트웨어는 전 세계에 있는 독립 프로그래머들로 구성된 팀을 통해 개발된다.
- **오픈 소스 진영** OSI, Open-Source Initiative 오픈 소스 소프트웨어를 증진하기 위한 그룹(http://www.opensource.com 참고)
- **옥타브 방법** OCTAVE(Operationally Critical Threat, Asset and Vulnerability Evaluation) method 카네기 멜론 대학교의 소프트웨어 엔지니어링 기관에서 개발한 시스템의 보안 위협 평가 기술
- **온라인** online 네트워크에 직접 연결되어 있는 컴퓨터의 상태
- **온라인 스페어** online spare '핫 스페어 디스크' 참고
- **온라인 인증서 상태 프로토콜** OCSP, Online Certificate Status Protocol 실시간으로 인증서를 검증하기 위한 프로토콜
- **온라인 트랜잭션 처리** OLTP, OnLine Transaction Processing 데이터베이스나 웹 서비스와 같이 소량의 데이터들을 담은 채 임의로 분산된 디스크들에 많은 수의 요청이 들어오는 유형의 시스템. 이러한 시스템은 디스크 스케줄링 알고리즘을 사용해 성능을 높일 수 있다.
- **온보드 장치** on-board device 컴퓨터 메인보드에 물리적으로 연결되는 장치
- **옵티마이저** optimizer 컴파일러의 일부로, 실행 효율을 향상하고 프로그램이 차지하는 저장 공간을 줄여주는 역할을 한다.
- **완전 순서** total ordering 모든 이벤트가 모든 프로세스에서 동일한 순서로 되는 것을 보장하는 것
- **외부 단편화** external fragmentation 가변 파티션 메모리 시스템에서 나타나는 현상으로, 프로세스를 보유하기에는 너무 작은 남은 공간이 메모리 전반에 산재하는 현상
- **외부 명칭** external name 한 모듈에서 정의된 심볼 중 다른 모듈에서 참조할 수 있는 것
- **외부 참조** external reference 한 모듈이 다른 모듈에 있는 외부 이름을 참조하는 것

- **요구 페이징** demand paging 프로세스가 명시적으로 참조할 때, 그 페이지를 메모리에 로드하는 기술
- **요구 페치 전략** demand fetch strategy 프로그램 부분이나 데이터를 프로세스에서 요청할 때 메모리에 로드하는 방법
- **요청 리스트** request list **(리눅스)** 보류 입출력 요청을 저장하는 구조체. 이 리스트는 탐색 시간을 줄여 처리량을 높이는 방향으로 정렬된다.
- **요청률** request rate 요청 빈도. 요청률이 높을수록 시스템 부하가 커진다. 디스크 요청률이 높은 시스템은 디스크 스케줄링을 통해 이점을 얻을 수 있다.
- **우선순위** priority 프로세스나 스레드의 중요도를 나타내며, 이들이 실행될 순서와 기간을 결정하는 데 사용한다.
- **우선순위 구매** purchase priority 시스템에서 더 높은 우선순위를 얻기 위해 추가 비용을 지불하는 일
- **우선순위 배열** priority array **(리눅스)** 실행 큐 내의 구조체로 동일한 우선순위를 가진 프로세스들을 저장한다.
- **우선순위 에이징** aging of priority 프로세스가 시스템에서 대기하는 시간에 근거해 우선순위를 증가시켜주는 일
- **운영체제** OS, Operating System 시스템의 자원을 관리하고 응용 프로그램이 적절하게 실행할 수 있게 서비스하는 소프트웨어. 운영체제는 하드웨어와 소프트웨어 자원을 모두 관리하기도 하며, 응용 프로그램 개발을 위한 인터페이스를 제공한다. 또한 안정적이고 보안성 있고, 반응성이 좋은 환경을 제공하며, 사용자들이 시스템 자원을 편리하게 활용할 수 있게 돕는다.
- **워드** word 시스템 프로세서가 한 번에 처리할 수 있는 비트 수. 데이터 계층에서 워드는 바이트보다 한 단계 상위 수준이다.
- **워크스테이션 로그인 스크립트** workstation login script 간단한 형태의 싱글 사인 온으로, 사용자는 자신의 워크스테이션에서 로그인하고 메뉴에서 응용 프로그램을 선택한다.
- **원거리 페이지 교체 전략** far page-replacement strategy 그래프 기반의 페이지 교체 전략으로, 프로그램의 참조 패턴을 분석해 교체할 페이지를 결정한다. 이 전략은 그래프에서 참조된 페이지들로부터 가장 멀리 떨어져 있고 최근 참조되지 않은 페이지를 교체 대상으로 선택한다.
- **원격 메소드 호출** RMI, Remote Method Invocation 자바 프로그래머들이 명시적으로 소켓을 작성하지 않고도 분산 시스템을 구현할 수 있도록 해준다.
- **원격 스풀링 통신 서브시스템** RSCS, Remote Spooling Communications Subsystem VM 구성 요소로, 분산 시스템에서 파일을 주고받을 수 있는 기능을 제공한다.
- **원격 참조 계층** RRL, Remote Reference Layer RMI에서 마셜링된 메시지를 클라이언트와 서버 사이에 보내기 위해 전송 계층과 일한다.
- **원격 프로시저 호출** RPC, Remote Procedure Call 한 컴퓨터에서 실행되는 프로세스가 다른 컴퓨터에서 실행되는 프로세스의 프로시저나 함수를 호출할 수 있게 해주는 것
- **원자적 브로드캐스트** atomic broadcast 시스템의 모든 메시지들이 각 프로세스마다 동일한 순서로 수신되는 것을 보장한다. 이는 '완전 순서 브로드캐스트' 또는 '합의 브로드캐스트'로도 알려져 있다.

- **원자적 연산** atomic operation 인터럽트되지 않고 전체가 실행되는 연산
- **원자적 트랜잭션** atomic transaction 전체가 완료되지 않으면 시스템의 상태에 영향을 미치지 않는 연산들의 집합
- **월드 와이드 웹** WWW, World Wide Web 하이퍼링크로 연결된 문서 집합으로, HTTP HyperText Transfer Protocol를 사용해 인터넷으로 접근할 수 있다. 문서들은 보통 HTML HyperText Markup Language이나 XML eXtensible Markup Language 등을 사용해 작성한다.
- **웜** worm 네트워크를 통해 전파되면서 파일들을 감염하는 실행 코드. 웜이 전파되는 데는 사용자의 동작이 필요하지 않으며 다른 프로그램이나 파일에 첨부될 필요도 없다.
- **웹 서비스** web service 인터넷을 통해 응용 프로그램이 통신하고 데이터를 주고받을 수 있게 해주는 서비스와 관련 표준. 웹 서비스는 이기종 플랫폼에서 여러 다른 언어로 작성된 요소들이 개방된 문자 기반 표준을 통해 통신하도록 한다. 이들을 인터넷에 있는 '즉시 사용 가능한' 소프트웨어 부품이라 할 수 있다.
- **웹 위조** web defacing 기관의 웹사이트를 악의적으로 수정하는 공격
- **위치** location **(파일)** 파일이 저장 장치나 시스템의 논리적인 파일 구성 안에 있는 주소
- **위치 결정 시간** positioning time 접근 시간에 지연 시간을 더한 것. 위치 결정 시간은 요청들을 정렬하기 위해 SPTF 전략에서 사용한다.
- **위치 독립 호출** location-independent call 호출이 실행되는 워크스테이션에 종속되지 않는 시스템 호출. 위치 독립 호출은 모든 워크스테이션에서 동일한 결과를 낸다.
- **위치 의존 호출** location-dependent call (스프라이트에서) 호출이 수행될 워크스테이션에 의존적인 시스템 호출. 위치 의존 호출은 워크스테이션이 달라질 때 다른 결과를 낸다.
- **위치 투명성** location transparency 접근 투명성에 기반해 분산 시스템에서 자원들을 접근하는 개체에 자원들의 위치를 감추는 것
- **윈도우 API** Windows API 마이크로소프트의 인터페이스로, 윈도우 환경에서 실행되는 응용 프로그램에서 사용한다. 윈도우 API는 프로그래머들이 운영체제 서비스를 호출하게 함으로써 이러한 연산을 수행하는 코드를 작성하지 않아도 되게 해준다. 또한 운영체제가 자신의 자원을 보호할 수 있게 해준다.
- **윈도우 관리자** window manager **(리눅스)** GUI 중 윈도우의 배치와 모양, 크기, 기타 속성을 제어하는 응용 프로그램
- **유니박 1** UNIVAC 1, UNIVersal Automatic Computer 마그네틱 저장소 테이프를 선보인 최초의 컴퓨터
- **유니코드** Unicode 국제적인 언어들을 지원하는 문자 집합으로, 인터넷과 다국어 응용 프로그램에 유용하게 쓰인다.
- **유닉스** UNIX 벨 연구소에서 C 언어로 개발한 운영체제
- **유형** type **(파일)** (실행 프로그램, 데이터, 혹은 디렉토리 같은) 파일의 용도 기술
- **유효 우선순위** effective priority **(리눅스)** 프로세스의 정적 우선순위에 우선순위를 더하거나 빼서 조절한 우선순위

- **응용 프로그래밍** application programming 운영체제에 서비스와 자원 사용을 요청해 (문서 작성, 웹 페이지 로딩 등) 필요한 임무를 수행하는 코드를 작성해 소프트웨어를 개발하는 일
- **응용 프로그래밍 인터페이스** API, Application Programming Interface 응용 프로그램이 운영체제나 라이브러리 모듈 등 하위 수준 시스템으로부터 서비스를 요청하도록 해주는 함수 집합
- **응용 프로그램 기반** application base 응용 프로그램을 개발하는 하드웨어와 운영체제 환경의 조합으로, 일단 응용 프로그램 기반이 확립되면 응용 프로그램 개발자가 다른 기반으로 옮겨가기 어렵다.
- **응용 프로그램 수준 게이트웨이** application-level gateway 패킷에 담긴 데이터로부터 네트워크를 보호하는 하드웨어나 소프트웨어. 만일 메시지가 바이러스를 포함하고 있으면, 게이트웨이가 막아서 의도된 수신자에게 전달되지 못하게 한다.
- **이기적인 라운드로빈 스케줄링** SRR(Selfish Round-Robin) scheduling 라운드로빈 스케줄링의 변형으로, 프로세스들의 에이징 속도가 다르다. 시스템에 유입된 프로세스들은 보류 큐에 놓이고 여기서 우선순위가 충분히 높아지면 활성 큐에 놓인다. 활성 큐에 놓인 프로세스들은 프로세서 시간을 위해 경쟁할 수 있다.
- **이동 투명성** migration transparency 객체가 한 위치에서 시스템의 다른 위치로 옮기는 것을 가리는 것. 예를 들어, 파일이 서버에서 다른 머신으로 이동하는 것 등이다.
- **이름 변경** rename **(파일)** 파일의 이름을 변경하는 연산
- **이름 있는 파이프** named pipe **(리눅스)** 디렉토리 트리에 의해 접근할 수 있는 파이프로, 부모/자식 관계가 아닌 프로세스들이 서로 통신할 수 있게 해준다. '파이프' 참고
- **이식성** portability 다양한 플랫폼에서 실행할 수 있는 소프트웨어의 속성
- **이식성 있는 운영체제** portable operating system 여러 가지 하드웨어 구성 환경에서도 작동하도록 설계된 운영체제
- **이전 발생 관계** happens before relation A, B가 동일한 프로세스에 속해 있고 A가 B보다 먼저 발생하면, 'A는 B 전에 발생한다' 고 말한다. 혹은 A는 메시지를 보내는 쪽이고 B는 그 메시지를 수신하는 쪽이다.
- **이중 간접 포인터** doubly indirect pointer (단일) 간접 포인터 블록의 위치를 가리키는 아이노드 포인터
- **이중 요소 인증** two-factor authentication 사용자를 인증하는 데 두 가지 수단을 사용하는 인증 기술. 예를 들면, 생체 인식과 패스워드 혹은 스마트 카드와 패스워드를 함께 확인하는 방식이다.
- **인과적 브로드캐스트** causal broadcast 한 프로세스로부터 모든 다른 프로세스에 메시지가 전송될 때, 모든 프로세스는 다른 프로세스로부터 새로운 메시지를 받기 전에 그 메시지를 받는 것이 보장된다.
- **인과적 순서** causal ordering 모든 프로세스가 인과적으로 종속적인 이벤트가 종속되는 메시지 이후에 발생함을 인식하도록 보장한다.
- **인과적 종속** causally dependent 이벤트 A가 발생할 때만 이벤트 B가 발생한다면, 이벤트 B는 이벤트 A에 인과적으로 종속된다.
- **인덱스 블록** index block 파일 데이터 블록을 가리키는 포인터 리스트를 담는 블록
- **인덱스를 사용한 순차적 파일 구성** indexed sequential file organization 레코드들을 각 레코드가 담은 키에 따라 논리적

순서로 정렬하는 파일 구성

- **인쇄 회로 기판** PCB, Printed Circuit Board 보드 여러 곳에 위치하는 장치들에 전기적 연결을 제공하는 하드웨어 부품
- **인위적 연속성** artificial contiguity 가상 머신 시스템에서 사용하는 기술로, 프로그램의 명령어와 데이터가 실제로는 부분들로 메인 메모리 여러 곳에 산재해 있을 때도 연속적으로 저장된 것처럼 보이게 함으로써 프로그래밍을 간단하게 해준다.
- **인증** authentication **(보안 트랜잭션)** 메시지 송신자와 수신자가 서로의 신원을 확인할 수 있게 하는 문제를 다룬다.
- **인증 서버 스크립트** authentication server script 중앙 서버를 통해 사용자를 인증하는 싱글 사인 온 구현으로, 사용자와 사용자가 접근하려고 하는 응용 프로그램 사이의 연결을 확립한다.
- **인증 철회 목록** CRL, Certificate Revocation List 취소되고 철회된 인증서의 목록. 인증서는 만료일 이전에 비밀키가 뚫릴 경우 취소/철회된다.
- **인증 헤더** AH, Authentication Header **(IPSec)** 패킷 송신자의 신원을 확인해주고, 패킷의 데이터가 전송 도중에 수정되지 않았음을 증명해주는 정보
- **인증기관** CA, Certificate Authority 금융 기관이나 VeriSign과 같은 신뢰할 수 있는 제3자로, 디지털 증명서를 발행하는 기관
- **인증기관 계층** certificate authority hierarchy 인증기관의 체인으로, 인증서와 인증기관들을 인증하는 루트 기관에서부터 시작된다.
- **인증서 저장소** certificate repository 디지털 인증서가 저장되는 위치
- **인터넷** internet 원격 통신과 월드 와이드 웹을 지원하는 네트워크 통신 근간으로, 인터넷에 연결된 각 컴퓨터는 원하는 서비스를 요청하기도 하고, 인터넷에 연결된 다른 컴퓨터에 서비스를 제공하기도 한다.
- **인터넷 키 교환** IKE, Internet Key Exchange **(IPSec)** IPSec에서 사용하는 키 교환 프로토콜로, 안전한 키 교환을 가능하게 해준다.
- **인터넷 프로토콜 보안** IPSec, Internet Protocol Security **(IPSec)** 전송 계층의 보안 프로토콜로 데이터 프라이버시, 무결성, 인증 기능을 제공한다.
- **인터럽트** interrupt 특정 이벤트가 발생한 사실을 알리는 하드웨어 신호. 인터럽트는 프로세서가 '인터럽트 처리기'라는 일련의 소프트웨어 명령어를 호출하게 한다.
- **인터럽트 벡터** interrupt vector 보호되는 메모리에 있는 배열로, 인터럽트 처리기들의 위치를 가리키는 포인터를 담고 있다.
- **인터럽트 비활성화(마스크)** disable(mask) interrupts 어떤 프로세스가 특정 유형의 인터럽트를 비활성화(마스킹)하면, 해당 유형의 인터럽트는 해당 프로세스에 전달되지 않는다. 인터럽트를 후에 전달할 수 있도록 큐에 저장하거나 프로세서에서 그냥 버린다.
- **인터럽트 처리기** interrupt handler 인터럽트에 대한 반응으로 실행되는 커널 코드

- **인터럽트 처리기 상단** top half of an interrupt handler **(리눅스)** 인터럽트 처리 코드 중 비선점 부분. 선점 가능한 하단 처리기로 실행을 전환하기 전에 승인을 위해 필요한 최소한의 작업을 수행한다.
- **인터럽트 처리기 하단** bottom half of an interrupt handler **(리눅스)** 인터럽트 처리 코드 부분 중 선점될 수 있는 부분
- **인터럽팅 클록(간격 타이머)** interrupting clock(interval timer) 특정 시간(이 시간 양을 퀀텀이라고 한다)이 지나면 인터럽트를 발생하는 하드웨어 장치. 인터럽팅 클록은 한 프로세스가 프로세서를 독점하는 일을 방지하는 등의 목적으로 사용한다.
- **인터페이스 정의 언어** IDL, Interface Definition Language RPC의 세부 사항을 명시하는 언어. RPC는 언어 독립적으로 인터페이스를 정의하고 분산 응용 프로그램들이 원격 컴퓨터의 프로시저를 투명하게 호출할 수 있게 해준다.
- **인터프리터** interpreter 기계어가 아닌 코드를 한 문장 번역해 (즉, 고급 언어 명령어를) 실행하는 응용 프로그램
- **일대일 맵핑** one-to-one mapping 각 사용자 수준 스레드가 한 커널 수준 스레드에 할당되게 하는 스레딩 모델
- **일반 메모리** normal memory **(리눅스)** 16MB를 넘어서는 물리 메모리 위치로, 커널이 가상 주소 공간에 직접 맵핑할 수 있다. 이 영역은 커널 데이터와 사용자 페이지를 저장하는 데 사용한다.
- **일반 보호 폴트** GPF, General Protection Fault **(IA-32 인텔 아키텍처)** 프로세스가 한 세그먼트를 참조할 때, 적절한 접근 권한을 갖지 않을 때 혹은 세그먼트 밖의 주소를 참조할 때 발생하는 폴트
- **일반 세마포어** general semaphore 카운팅 세마포어 참조
- **일시 정지 블록 상태** suspendedblocked state 프로세스가 블록되어 있을 때 일시 정지된 상태. 이러한 프로세스를 재시작하면 다시 블록 상태로 돌아온다.
- **일시 정지 상태** suspended state 일시 정지 블록 혹은 일시 정지 준비 상태로, 이때 프로세스는 소멸되지 않지만 프로세서를 얻으려고 경쟁하는 것이 무기한 연기된다. 역사적으로 이 연산은 시스템 운영자가 시스템의 부하를 수동으로 조정할 때나 시스템이 다운될 위험에 대처할 때 사용되어왔다.
- **일시 정지 준비 상태** suspendedready state 프로세스가 준비 상태에 있을 때 일시 정지된 상태. 이러한 프로세스를 재시작하면 다시 준비 상태로 돌아온다.
- **일시 정지/재시작** suspend/resume 프로세스를 멈추고 그 상태를 저장하고, 자원을 다른 프로세스들에 반납한 후 다른 프로세스들이 이들을 해제할 때 다시 할당해주는 방법
- **읽기** read **(파일)** 파일의 데이터 항목을 프로세스에 입력하는 연산
- **읽기 접근** read access **(가상 메모리)** 프로세스가 페이지나 세그먼트의 데이터를 읽을 수 있도록 하는 접근 권한
- **읽기 접근** read access **(파일)** 파일을 읽을 수 있게 허용하는 권한
- **읽기-수정-쓰기 메모리 연산** RMW(Read-Modify-Write) memory operation 변수의 내용을 원자적으로 읽고, (읽은 것을 기반으로) 내용을 변경한 후 새로운 값을 메모리에 쓰는 연산. 이러한 연산은 원자적 연산을 제공해 상호 배제 알고리즘을 간단하게 해준다.
- **읽기-수정-쓰기 사이클** read-modify-write cycle **(RAID)** 스트라이프를 읽고, 내용과 패리티를 수정한 후 해당 스트라이프를 배열에 쓰는 연산. 패리티를 사용하는 RAID 시스템에서 각각의 쓰기 요청에 관해 수행된다. 몇몇 시스

템은 스트립을 캐시하거나 패리티 정보를 주기적으로 갱신하는 방법으로 이 연산의 비용을 줄이기도 한다.

- **읽기-쓰기 헤드** read-write head 무빙 헤드 디스크 구성 요소로, 디스크 표면을 떠다니며 디스크가 움직일 때 비트 열을 읽고 쓴다.
- **임계 영역** critical section(critical region) 공유 자원에 대해 연산을 수행하는 코드 부분(예를 들면, 공유되는 변수에 데이터를 쓰는 것). 프로그램이 정확하게 동작하려면 한 번에 최대 한 스레드만 임계 영역 안에서 수행되어야 한다.
- **임베디드 시스템** embedded system 제한된 자원과 특별한 하드웨어를 갖춘 소형 컴퓨터로, PDA나 휴대폰에 사용하는 컴퓨터 시스템
- **임시 파일 시스템** tmpfs, temporary file system **(리눅스)** ramfs와 비슷하지만 사용 전에 포맷할 필요가 없다. 즉, 시스템인 대부분의 파일 시스템에서 요구하는 구성 오버헤드 없이도 tmpfs에 파일을 저장할 수 있다.
- **임의 접근 메모리** RAM, Random Access Memory 어떤 순서로든 해당 내용에 접근할 수 있는 메모리
- **임의 탐색 패턴** random seek pattern 디스크 표면에 임의적으로 분산된 실린더의 데이터들을 요청하는 것. 선착 우선 알고리즘은 임의 탐색 패턴을 일으키며 반응 시간이 늦고 처리량이 적다.
- **임의 페이지 교체** RAND(random) page replacement 임의적인 페이지 교체 전략으로, 메인 메모리의 각 페이지는 교체될 확률이 동일하다. 이 전략은 공평하고 오버헤드도 적지만, 미래의 페이지 사용을 예측하는 것과는 거리가 멀다.
- **입자성 비트** granularity bit **(IA-32 인텔 아키텍처)** 프로세서가 20비트 세그먼트 한계로 명시된 각 세그먼트 크기를 어떻게 해석할지를 결정하는 비트. 세그먼트는 비트가 'off'면 1바이트에서 1MB, 비트가 'on'이면 4KB에서 4GB 범위다.
- **입출력 관리자** I/O manager 운영체제 구성 요소로, 입출력 요청을 받고 해석해 수행하는 기능을 담당한다.
- **입출력 완료 인터럽트** I/O completion interrupt 어떤 장치가 입출력 작업을 완료할 때 발생하는 메시지
- **입출력 요청 병합** merge I/O requests **(리눅스)** 디스크에서 인접한 위치로의 입출력 요청 둘을 한 요청으로 병합하는 일
- **입출력 제어 시스템** IOCS, Input/Output Control System 현대 운영체제의 선구자로, 프로그래머들에게 입출력을 수행할 수 있는 기본적인 기능 집합을 제공한 시스템
- **입출력 중심** I/O-bound 프로세서를 잠시 사용해 입출력 요청을 생성한 후 프로세서를 반환하는 프로세스나 작업의 성격
- **입출력 채널** I/O channel 메인 프로세서에 독립적으로 장치 입출력을 처리하는 구성 요소
- **자격** capability 객체에 대한 토큰을 부여함으로써 주체의 접근 권한을 할당하는 보안 메커니즘. 이는 관리자가 미세 단위 접근 제어를 지정하고 강화할 수 있게 해준다.
- **자격** capability **(접근 제어 메커니즘)** 객체에 대한 권한을 주체에 주는 토큰. 지참인이 스포츠 이벤트에 대한 접근을 얻는 데 사용하는 티켓에 해당한다.
- **자격 증명** credential 사용자 이름 등과 같은 사용자 식별자와 해당 식별자를 증명할 패스워드와 같은 것

- **자바** Java 썬 마이크로시스템즈에서 개발한 객체지향 프로그래밍 언어로, 가상 머신에서 실행하게 함으로써 이식성을 높여준다.
- **자바 가상 머신** JVM, Java Virtual Machine 자바 프로그램을 네이티브 머신 언어로 다시 컴파일할 필요 없이 이기종 아키텍처에서 실행할 수 있게 해주는 가상 머신이다. 자바 가상 머신은 응용 프로그램의 이식성을 높이고, 프로그래머가 각 아키텍처마다 특정한 세부 항목을 고려해야 하는 부담을 덜어준다.
- **자발적인 페이지 해제** voluntary page release 프로세스가 더는 필요하지 않은 페이지 프레임들을 명시적으로 해제하는 것. 이는 사용되지 않는 페이지 프레임이 프로세스에 할당되어 있는 시간을 줄여주고, 더 많은 여유 메모리를 만들어준다.
- **자식 프로세스** child process 부모 프로세스로부터 생성된 프로세스. 자식 프로세스는 프로세스 계층에서 자신의 부모 프로세스보다 한 수준 아래에 있다. 유닉스 시스템에서는 fork라는 시스템 호출을 사용해 자식 프로세스를 생성한다.
- **자원 유형** resource type 공통 작업을 수행하는 자원끼리 그룹화한 것
- **자원 할당 그래프** resource allocation graph 시스템의 프로세스와 자원을 보여주는 그래프. 프로세스에서 자원으로 향하는 화살은 프로세스가 해당 자원을 요청함을 나타낸다. 자원에서 나와 프로세스를 가리키는 화살은 자원이 해당 프로세스에 할당되었음을 나타낸다. 이러한 그래프는 시스템에 교착 상태가 존재하는지, 이와 연관된 프로세스와 자원이 무엇인지 판단하는 데 도움이 된다.
- **자유 재량 접근 제어** discretionary access control 파일 소유자가 해당 파일에 대한 사용자 접근 권한을 지정할 수 있게 하는 접근 제어 정책
- **작업** job 컴퓨터가 수행하는 일
- **작업** task **(리눅스)** 리눅스에서의 사용자 실행 문맥(즉, 프로세스나 스레드)
- **작업 간 전환** job-to-job transition 단일 스트림 배치 처리 시스템에서 한 작업을 시스템에서 제거하고, 다음 작업을 로드해 실행 준비를 하는 사이 작업을 실행할 수 없는 기간
- **작업 디렉토리** working directory 사용자가 직접 접근할 수 있는 파일을 담은 디렉토리
- **작업 스케줄링** job scheduling '고수준 스케줄링' 참고
- **작업 스트림 프로세서** job stream processor 단일 스트림 배치 처리 시스템의 개체로, 작업 간 전환을 제어한다.
- **작업 제어 언어** job control language 작업 스트림 프로세서에서 해석한 명령어로, 단일 스트림 배치 처리 시스템에서 다음 작업을 위한 준비를 정의하고 도와준다.
- **작업 집합** working set 프로그램이 선호하는 페이지들의 부분 집합. 작업 집합 윈도우 w가 주어졌을 때, 프로세스의 작업 집합 $W(t, w)$는 $t-w$에서부터 t까지의 프로세스 시간 동안 참조한 페이지들을 가리킨다.
- **작업 집합 윈도우 크기** working set window size 시스템이 프로세스의 작업 집합에 해당하는 페이지를 알아내기 위해 얼마만큼 과거 시간까지 아울러 계산할 것인지를 결정하게 하는 값
- **작업자 스레드** worker thread 스레드 풀에 있는 스레드로, 작업자 스레드는 해당 스레드 풀을 생성한 프로세스 내의 모든 사용자 스레드에 맵핑될 수 있다.

- **장기 스케줄링** long-term scheduling '고수준 스케줄링' 참고
- **장애 내구성** fault tolerance 소프트웨어나 하드웨어의 오류를 보완해 정상적으로 동작할 수 있게 하는 운영체제의 능력
- **장치 독립성** device independence 응용 프로그램이 저장 장치의 이름 대신 심볼릭 이름을 사용해 파일을 참조할 수 있게 하는 파일 속성
- **장치 클래스** device class 유사한 기능을 수행하는 장치들의 그룹
- **장치 특수 파일** device special file **(리눅스)** /dev 디렉토리에 있는 개체로, 특정 장치에 접근할 수 있게 해준다.
- **재배치** relocating 프로그램 코드와 데이터의 주소를 조정하는 과정
- **재배치 가능한 로딩** relocatable loading 요청된 메모리 블록의 위치를 기반으로 로드 모듈에 있는 상대 주소를 절대 주소로 변환하는 로딩 방식
- **재배치 투명성** relocation transparency 객체의 재배치를 그와 통신하는 다른 객체로부터 감춘다.
- **재시작** resume 일시 정지 상태에 있는 프로세스를 제거하는 일
- **재진입 코드** reentrant code 사용 중에 수정되지 않아 프로세스나 스레드 사이에 공유될 수 있는 코드
- **저널링 파일 시스템** JFS, Journaling File System '로그 구조 파일 시스템' 참고
- **저수준 스케줄링** low-level scheduling 어떤 프로세스가 프로세서 제어를 얻을지 결정하는 일
- **저장 장치 포맷** format a storage device 내용을 검사하고 저장 관리 메타데이터를 쓰는 등의 연산을 수행해 장치를 파일 시스템에 사용할 수 있게 준비하는 일
- **적응 메커니즘** adaptive mechanism 시스템의 동작 방식이 변할 때 이에 반응해 조절하는 제어 메커니즘
- **전송 계층** transport layer **(RMI)** RMI에서 RRL과 협력해 마셜링된 메시지를 클라이언트와 서버 사이에 보내는 계층
- **전송 시간** transmission time 데이터 레코드가 읽기/쓰기 헤드를 지나는 데 걸리는 시간
- **전역 기술자 테이블** GDT, Global Descriptor Table **(IA-32 인텔 아키텍처)** 세그먼트 맵 테이블로, 프로세스 세그먼트나 지역 기술자 테이블 세그먼트를 위한 맵핑 정보를 담고 있다.
- **전역 최소 최근 사용 페이지 교체** gLRU(global Least-Recently-Used) page replacement 시스템 전체에서 가장 오랫동안 참조되지 않은 페이지를 교체 대상으로 선정하는 전역적인 페이지 교체 전략. LRU의 라운드로빈 스케줄링의 변형은 시스템이 대규모 루핑 참조 패턴을 보이게 해 성능을 떨어뜨릴 수 있다. gLRU의 SEQ(순차) 변형은 루핑 참조 패턴을 탐지할 때 가장 적게 사용된 페이지를 교체함으로써 성능 향상을 꾀한다.
- **전용 자원** dedicated resource 한 번에 한 프로세스에서만 사용할 수 있는 자원. '순차적으로 재사용 가능한 자원'이라고도 한다.
- **전위 암호** transposition cipher 문자들의 순서가 뒤바뀌는 암호화 기술. 예를 들어, 'security'라는 단어에서 's'로

시작해 매 두 번째 글자가 앞부분을 형성하고, 나머지 글자가 뒷부분을 형성한다면, 'scrt euiy'로 암호화될 것이다.

- **전이 상태** transition state **(윈도우 XP)** 스레드가 대기를 완료했지만 (자신의 커널 스택이 메모리에서 페이징 아웃되어) 아직 실행될 준비는 되지 않았음을 나타내는 스레드 상태
- **절대 경로** absolute path 루트 디렉토리에서 시작하는 경로
- **절대 로딩** absolute loading 로더가 프로그래머나 컴파일러가 지정한 메모리 주소에 프로그램을 위치시키는 기술
- **절차적 프로그래밍 언어** procedural programming language 객체(명사)보다는 함수(동사)에 초점을 둔 프로그래밍 언어
- **접근 격리 메커니즘** AIM, Access Isolation Mechanism **(멀틱스)** 멀틱스 시스템에서 강제적인 접근 제어 메커니즘을 구현한 것
- **접근 권한** access right 다양한 주체가 다양한 객체에 어떻게 접근할 수 있는지를 정의한다. 주체는 사용자나 프로세스, 프로그램 혹은 다른 개체가 될 수 있다. 객체는 정보 은닉을 하는 개체로, 디스크, 프로세스, 메인 메모리 등에 대응하는 물리적 객체일 수도 있고, 자료 구조, 프로세스 혹은 서비스와 같은 추상적인 객체일 수도 있다. 주체는 또한 시스템의 객체로 간주할 수도 있다. 한 주체가 다른 주체에 접근할 권한을 가질 수도 있다.
- **접근 방법** access method 파일 시스템이 파일에 접근하는 데 사용하는 방법. '대기 접근 방법'과 '기본 접근 방법' 참고
- **접근 비트** accessed bit '참조 비트' 참고
- **접근 제어 매트릭스** access control matrix 시스템의 주체를 행에 나열하고 이들이 접근 권한을 얻은 객체를 열에 나열하는 매트릭스. 매트릭스의 각 셀은 주체가 객체에 대해 수행할 수 있는 동작을 명시한다. 접근 제어 매트릭스는 내용이 너무 드문드문하므로 잘 구현되지 않는다.
- **접근 제어 모드** access control mode 읽기, 쓰기, 실행, 추가 등의 권한 집합으로, 페이지나 세그먼트를 접근할 방법을 결정한다.
- **접근 제어 목록** ACL, Access Control List 어떤 객체에 관해 어떤 주체가 부여받은 각 접근 권한을 나타내는 엔트리를 저장하는 목록. 접근 제어 목록은 접근 제어 매트릭스보다 공간을 적게 소비한다.
- **접근 제어 목록** ACL, Access Control List **(멀틱스)** 멀틱스 시스템에서 자유재량 접근 제어 메커니즘을 구현한 것
- **접근 제어 속성** access control attribute **(리눅스)** 특정 자원에 접근하려는 프로세스들의 접근 권한을 지정한다.
- **접근 투명성** access transparency 분산 시스템에서 컴퓨터 간의 통신을 가능하게 하는 네트워킹 프로토콜의 세부 사항을 감추는 것
- **접근성** accessibility **(파일)** 어떤 사용자가 파일 데이터에 접근할 수 있는지 제한하는 파일 속성
- **정도** degree **(릴레이션)** 관계형 데이터베이스에서 테이블(릴레이션)의 속성 수
- **정보 은닉** information hiding 객체 외부로부터 객체 안에 있는 데이터를 직접 접근하는 일을 방지해, 더 신뢰성 있는 소프트웨어 시스템을 개발할 수 있도록 돕는 소프트웨어 아키텍처 기술

- **정적 RAM** SRAM, Static RAM 재생할 필요가 없고, 전원이 공급되는 한 데이터를 보유할 수 있는 RAM
- **정적 분석** static analysis 크랙커가 응용 프로그램을 손상할 때 시스템 침입을 탐지하는 방법
- **정적 실시간 스케줄링 알고리즘** static real-time scheduling algorithm 시간적 제약을 통해 프로세스가 실행되기 전에 고정된 우선순위를 부여하는 스케줄링 정책
- **정적 우선순위 수준** static priority level **(리눅스)** 프로세스가 생성될 때 스케줄링 우선순위를 결정하기 위해 부여되는 정수 값
- **정책 생성 기관** policy creation authority 디지털 인증서를 위한 정책을 지정하는 기관
- **제어 프로그램** CP, Control Program **(VM)** VM 구성 요소로 물리 머신을 실행하고 가상 머신을 위한 환경을 생성한다.
- **제한된 알고리즘** restricted algorithm 송신자와 수신자가 동일한 암호화 알고리즘을 사용해 비밀을 지킨다는 것에 의존하는 보안 알고리즘
- **조각 모음** defragmentation 파일 조각들을 이동시켜 디스크의 연속적인 블록에 위치하도록 하는 것. 이 방법으로 순차적으로 파일을 읽고 쓸 때 접근 시간을 줄여준다.
- **조건 변수** condition variable 특수한 변수로, 값과 관련 큐를 갖는다. 스레드가 모니터 안에 있는 조건 변수를 대기할 때는, 모니터를 나오고 조건 변수의 큐에 놓인다. 스레드는 다른 스레드가 신호를 보낼 때까지 큐에서 대기한다.
- **조인** join **(데이터베이스)** 테이블들을 결합하는 연산
- **종료 상태** terminated state **(윈도우 XP)** 스레드가 실행을 마친 것을 나타내는 스레드 상태
- **종료 정리 작업** termination housekeeping 상호 배제 알고리즘의 경우, 운영체제에서 수행하는 작업으로, 상호 배제를 위반하지 않으면서도 어떤 스레드가 자신의 임계 영역에서 작업을 마쳤다면 다른 스레드가 작업을 진행할 수 있게 하는 일
- **주 버전 번호** major version number **(리눅스)** 중요한 리눅스 배포를 식별하게 하는 고유한 값
- **주 스레드** primary thread 프로세스 생성 시에 생성되는 스레드로, 실행의 메인 흐름(스레드)이라고도 한다. 주 스레드가 반환하면, 해당 프로세스도 종료한다.
- **주 실행 스레드** main thread of execution 프로세스 생성 시에 생성된 스레드로, 주 스레드라고도 한다.
- **주 장치 식별 번호** major device identification number **(리눅스)** 특정 장치 클래스에서 한 장치를 고유하게 식별하는 값. 커널은 이 값을 사용해 장치의 드라이버를 결정한다.
- **주기적 실시간 프로세스** periodic real-time process 정해진 시간 간격마다 규칙적으로 계산을 수행하는 실시간 프로세스
- **주소 공간** address space 한 프로세스가 참조하는 메모리 위치의 집합
- **주소 바인딩** address binding 프로그램 데이터와 명령어에 메모리 주소를 할당하는 일

- **주소 버스** address bus 버스의 일부분으로, 데이터가 전송되는 메모리 위치를 지정한다.
- **주소 변환 맵** address translation map 가상 주소를 대응하는 실제 메모리 주소로 맵핑하는 일을 돕기 위해 사용하는 테이블
- **죽음의 포옹** deadly embrace '교착 상태' 참고
- **준비 리스트** ready list 모든 '준비' 상태 프로세스를 담은 커널 자료 구조. 준비 리스트는 대체로 프로세스 스케줄링 우선순위에 따라 정렬된다.
- **준비 상태** ready state 프로세서를 할당받을 수 있는 프로세스(또는 스레드)의 상태
- **준비(실행 가능) 상태** ready(runnable) state 실행 상태가 되어 프로세서를 차지해 실행할 수 있는 스레드 상태. 윈도우 XP에서는 준비 상태 스레드가 즉시 대기 상태로 변하고, 여기서 다시 실행 상태로 변한다.
- **중간 수준 스케줄링** intermediate-level scheduling 어떤 프로세스를 저수준 스케줄러에 보내 프로세서를 얻기 위해 경쟁하게 할 것인지 결정하는 일
- **중간 코드 생성기** intermediate code generator 컴파일 과정의 한 단계로, 파서로부터 입력을 받고 옵티마이저에 명령어 스트림 결과물을 보내준다.
- **중단** abort 프로세스가 완료되지 않은 상태에서 종료하는 것. 또한 IA-32 명세에서는 프로세스가 복구할 수 없는 오류를 가리키기도 한다.
- **중량 프로세스** HWP, HeavyWeight Process 전통적인 프로세스로, 하나 혹은 여러 스레드를 포함할 수 있다. 프로세스를 중량이라고 표현하는 이유는 프로세스 생성 시마다 자제적인 주소 공간을 할당받아야 하기 때문이다.
- **중복성** redundancy 고유한 자원들을 중복으로 유지해 고장 나는 경우 복구를 가능하게 하는 기술
- **중앙 교착 상태 탐지** central deadlock detection 분산 교착 상태 탐지 전략으로, 한 사이트가 시스템 전체의 TWFG를 모니터링하는 것을 전담한다. 프로세스가 자원을 요청하거나 반납할 때마다 중앙에 있는 사이트에 이를 알린다. 그 사이트는 계속적으로 전체 TWFG가 순환 고리를 형성하는지 확인한다.
- **중앙 이동 서버** central migration server 스프라이트 분산 운영체제의 워크스테이션으로, 한가한 워크스테이션들에 대한 정보를 유지한다.
- **중앙 처리 장치** CPU, Central Processing Unit 컴퓨터의 일반적인 계산을 담당하는 프로세서
- **중첩** overlay 실행될 메인 메모리보다 큰 프로그램을 가능하게 해주는 개념. 프로그램은 작은 부분들로 나뉘며, 메모리에 동시에 존재하지 않아도 된다. 이러한 프로그램의 한 부분을 가진 것을 중첩이라고 한다.
- **즉시 대기 상태** standby state **(윈도우 XP)** 프로세서를 할당받아 실행되기로 선택된 스레드의 상태
- **즉시 쓰기 캐싱** write-through caching 캐시된 데이터가 수정될 때마다 데이터를 디스크 캐시 버퍼와 디스크 모두에 쓰는 기술. 이 기술을 사용하면 시스템이 요청들을 묶어서 처리할 수는 없지만 시스템이 다운될 경우 데이터가 불일치하게 되는 문제를 예방해준다.
- **증분 백업** incremental backup 파일 시스템에서 마지막 백업 이후 변경된 데이터만 백업하는 기술

- **지역 (메모리)** zone (memory) **(리눅스)** 물리 메모리 영역으로, 리눅스는 메인 메모리를 하위, 중간, 상위 지역으로 나누어 시스템 아키텍처의 제약 사항에 따라 메모리를 할당한다.
- **지역 기술자 테이블** LDT, Local Descriptor Table **(IA-32 인텔 아키텍처)** 세그먼트 맵 테이블로, 프로세스 세그먼트를 위한 맵핑 정보를 담고 있다. 시스템은 8,191개 LDT를 보유할 수 있고, 각 LDT는 8,192개 엔트리를 보유할 수 있다.
- **지역 할당자** zone allocator 지역에서 페이지들을 할당하는 메모리 서브시스템
- **지역성** locality 이벤트들이 공간이나 시간적으로 가까이 집중되는 경험적으로 관찰된 현상이다. 메모리 주소 패턴에 적용해보면, 공간적 지역성은 프로세스가 특정 주소 위치를 접근하면 그 근처의 주소도 접근할 확률이 높다는 것이다. 시간적 지역성은 프로세스가 특정 주소를 접근하면, 곧 다시 참조할 가능성이 크다는 것이다.
- **지연 시간** latency **(프로세스 스케줄링)** 작업이 서비스를 받기 전에 시스템에 머물면서 보내는 시간
- **지연된 취소** deferred cancellation **(POSIX)** 취소 모드 중 스레드가 자신이 취소 신호를 받았는지 명시적으로 확인한 후에야 종료하는 모드
- **지정 사용자** specified user **(파일 접근 제어)** 파일의 소유자 외에 해당 파일을 사용할 수 있는 개별 사용자의 ID
- **직렬 포트** serial port (키보드나 마우스와 같이) 한 번에 한 비트를 전송하는 장치들에 대한 인터페이스
- **직접 맵핑** direct mapping 주소 변환 메커니즘으로, 위치 기반 주소 방식의 메모리에 저장된 테이블의 인덱스를 사용해 가상 주소를 물리 주소에 맵핑한다.
- **직접 메모리 접근** DMA, Direct Memory Access 전송이 완료될 때 프로세서에 인터럽트를 보내는 컨트롤러를 통해 데이터를 메인 메모리에서 장치로 전송하는 방법. DMA를 사용한 입출력 전송은 프로세서가 각 바이트나 워드 데이터를 관장할 필요가 없기 때문에 프로그래밍된 입출력이나 인터럽트 방식 입출력보다 효율적이다.
- **직접 입출력** direct I/O 커널의 버퍼 캐시를 사용하지 않고 입출력을 수행하는 기술. 이 기술은 자신에 맞게 고유의 버퍼 캐시를 갖는 데이터베이스 응용 프로그램이 메모리를 더 효율적으로 활용할 수 있게 해준다.
- **직접 파일 구성** direct file organization 레코드가 직접 접근 저장 장치 DASD, Direct Access Storage Device에 있는 물리적 주소를 통해 직접(임의) 접근할 수 있는 파일 구성 기술
- **진입 허가 스케줄링** admission scheduling '고수준 스케줄링' 참고
- **질의 언어** query language 사용자들이 데이터베이스를 검색하여 특정 기준을 충족하는 데이터를 찾는 데 사용하는 언어
- **차선 적합 메모리 배치 전략** next-fit memory placement strategy 최초 적합 메모리 배치 전략의 변형으로, 여유 메모리 공간을 찾기 위해 이전에 검색이 끝났던 곳에서부터 시작해 다음 번 탐색을 시작한다.
- **참조 비트** referenced bit 페이지가 최근에 참조된 사실이 있는지 나타내는 페이지 테이블 엔트리 비트
- **처리 시간** turnaround time 작업 요청 후 결과를 반환받을 때까지 소요되는 시간
- **처리량** throughput 단위 시간 안에 실행하는 작업량. 단위 시간당 완료되는 프로세스의 수로 측정할 수 있다.

- **철학자들의 만찬** Dining Philosophers 다익스트라가 제기한 전통적인 문제로, 병행 프로그래밍에 내재된 교착 상태와 무기한 연기 문제를 묘사한다. 철학자 N명과 포크 n개가 있고, 철학자들은 먹고 생각하는 일을 반복한다. 프로그래머는 각 철학자가 양 옆에 놓인 포크를 획득해 먹을 수 있도록 해야 한다.
- **체이닝** chaining 인덱스를 사용한 불연속적 할당 기술로, 마지막 몇 개 엔트리를 남겨두었다가 더 많은 인덱스 블록을 가리키게 한다. 해당 블록은 또한 데이터 블록을 가리킨다. 체이닝 기법은 인덱스 블록이 블록 몇 개를 가로질러 큰 파일을 참조할 수 있게 해준다.
- **체이닝** chaining **(해시 테이블)** 해시 테이블의 충돌을 해결하기 위한 기술. 유일한 각 항목을 자료 구조(대체로 연결 리스트)에 놓는다. 해시 테이블에서 충돌이 발생하는 위치에는 이 자료 구조를 가리키는 포인터가 담긴다.
- **체크포인트** checkpoint **(트랜잭션 로깅)** 로그에서 어떤 트랜잭션이 영속적인 저장소에 전송되었는지 나타내는 표식. 시스템은 파일 시스템의 상태를 결정하기 위해 마지막 체크포인트부터 트랜잭션을 재적용하면 된다. 이는 로그 처음부터 시작하여 모든 트랜잭션을 재적용하는 것보다 훨씬 빠르다.
- **체크포인트** checkpoint **(교착 상태 복구)** 프로세스가 중도에 종료될 경우 복구를 돕기 위해 시스템 상태를 기록하는 것(즉, 교착 상태 복구를 수행하기 위한 것)
- **체크포인트/롤백** checkpoint/rollback 교착 상태 복구 방법으로, 마지막 체크포인트 이후, 종료시킨 프로세스의 모든 동작(트랜잭션)을 취소하는 일
- **초기화 상태** initialized state **(윈도우 XP)** 운영체제에서 스레드를 생성한 상태
- **최고 응답률 우선** HRRN, Highest-Response-Ratio-Next 프로세스의 서비스 시간과 프로세스가 대기한 시간에 근거해 우선순위를 할당하는 스케줄링 정책
- **최근 미사용 페이지 교체** NUR(Not-Used-Recently) page replacement 적은 오버헤드로 LRU와 비슷한 기능을 하는 페이지 교체 전략. 참조 비트와 수정 비트를 사용해 교체 페이지를 선정한다. NUR은 먼저 최근 참조되지 않고 수정되지 않은 페이지를 찾는다. 그런 페이지가 없을 때는 최근 참조되지 않은 수정된 페이지를 선택한다. 그 다음에는 최근 참조되었고, 수정되지 않은 페이지나 최근 참조된 페이지를 선택한다.
- **최단 데드라인 우선** EDF, Earliest-Deadline-First 데드라인이 임박한 프로세스에 프로세서를 할당해주는 스케줄링 알고리즘
- **최단 탐색 시간 우선 디스크 스케줄링** SSTF(Shortest-Seek-Time-First) disk scheduling 읽기/쓰기 헤드의 현재 실린더에 가장 가까운(따라서 탐색 시간이 가장 짧게 걸리는) 요청을 선택하는 디스크 스케줄링 전략. 평균 탐색 시간을 줄임으로써 SSTF는 FCFS보다 처리량을 높이고, 적절한 수준의 부하 조건에서 평균 반응 시간은 줄인다. 한 가지 심각한 약점은 가장 안쪽과 바깥쪽 실린더를 차별하기 때문에 반응 시간의 변량이 크다는 점이다. 극단적인 경우에는 읽기/쓰기 헤드에서 멀리 떨어진 요청들이 무기한 연기될 수도 있다.
- **최단 프로세스 우선 스케줄링** SPF(Shortest-Process-First) scheduling 비선점형 스케줄링 방식으로, 스케줄러는 완료 시까지 예상 실행 시간이 가장 짧은 프로세스를 선택해 실행한다.
- **최대 필요치** maximum need **(다익스트라의 은행원 알고리즘)** 다익스트라의 은행원 알고리즘에서 프로세스가 실행하는 데 필요한 (특정 유형의) 최대 자원 수

- **최소 권한의 원칙** principle of least privilege 사용자들이 위임한 작업을 수행하는 데 필요한 만큼만 권한을 부여받아야 한다는 자원 접근 정책
- **최소 사용 빈도 페이지 교체** LFU(Least-Frequently-Used) page replacement 가장 적게 참조된 페이지를 교체 대상으로 선정하는 페이지 교체 전략. LFU는 구현하기 쉽지만 일반적으로 미래의 페이지 사용을 잘 예측하지는 못한다.
- **최소 여유 시간 우선 스케줄링** minimum-laxity-first 프로세서를 가장 적게 사용해 완료할 수 있는 프로세스에 높은 우선순위를 부여하는 스케줄링 정책
- **최소 위치 결정 시간 우선 디스크 스케줄링** SPTF(Shortest-Positioning-Time-First) disk scheduling 위치 결정 시간이 가장 짧은 요청을 다음 서비스 대상으로 선택하는 디스크 스케줄링 전략. SPTF는 SSTF와 비슷하게 처리량이 많고 평균 반응 시간이 짧으며, 가장 안쪽이나 바깥쪽 실린더로의 요청을 무기한 연기할 위험이 있다.
- **최소 잔여 시간 우선 스케줄링** SRT(Shortest-Remaining-Time) scheduling SPF의 선점형 방식으로, 스케줄러는 완료 시까지 예상 실행 시간이 가장 적게 남은 프로세스를 선택한다.
- **최소 접근 시간 우선 디스크 스케줄링** SATF(Shortest-Access-Time-First) disk scheduling 최소의 접근 시간(즉, 위치 결정 시간 + 전송 시간)을 요구하는 요청을 다음 서비스 대상으로 선택하는 디스크 스케줄링 전략. SATF는 SPTF보다 처리량이 높지만 작은 요청들이 계속 들어올 때 큰 요청이 무기한 연기될 수 있고, 중간 실린더에 대한 요청 때문에 가장 안쪽이나 바깥쪽 실린더에 대한 요청이 무기한 연기될 위험이 있다.
- **최소 지연 시간 우선 디스크 스케줄링** SLTF(Shortest-Latency-Time-First) disk scheduling 대기하는 모든 요청들을 점검해 회전 지연이 가장 적게 걸리는 요청을 먼저 서비스하는 디스크 스케줄링 전략. 이 전략은 이론적인 최적의 알고리즘에 가깝고 구현하기도 비교적 쉽다.
- **최소 최근 사용 페이지 교체** LRU(Least-Recently-Used) page replacement 가장 오랫동안 참조되지 않은 페이지를 교체하는 페이지 교체 전략. LRU는 일반적으로 미래의 페이지 사용 패턴을 잘 예측하지만, 상당한 오버헤드를 발생시킨다.
- **최악 적합 전략** worst-fit strategy 메모리 배치 전략 중 들어오는 작업을 메모리의 가장 큰 홀에 배치하는 전략
- **최적 페이지 교체** OPT(optimal) page replacement 가장 먼 미래까지 사용되지 않을 페이지를 교체 대상으로 선정하는 방식으로, 최적의 전략임이 증명되었지만, 실현은 불가능하다.
- **최적합 메모리 배치 전략** best-fit memory placement strategy 메모리 배치 전략으로, 들어오는 작업을 보유할 수 있는 가장 작은 홀에 작업을 배치한다.
- **최초 적합 메모리 배치 전략** first-fit memory placement strategy 메모리 배치 전략으로, 들어오는 프로세스를 보유할 만큼 여유가 있는 첫 번째 홀에 배치한다.
- **추가 전용 파일 속성** append-only file attribute **(리눅스)** 파일 속성 중 사용자가 기존 파일 내용에 데이터 '추가'만 할 수 있도록 제한하는 파일 속성
- **추가 접근** append access 프로세스가 세그먼트의 끝에 추가적인 정보를 쓸 수는 있지만, 기존 내용을 수정할 수는 없게 하는 접근 권한
- **충돌** collision **(해시 테이블)** 해시 함수가 두 개의 다른 항목을 해시 테이블의 동일 위치에 맵핑하는 상황. 몇몇 해

시 테이블은 이러한 충돌 문제를 해결하려고 체이닝 기법을 사용한다.

- **침입 탐지 시스템** IDS, Intrusion Detection System 네트워크와 응용 프로그램 로그 파일들을 모니터링하는 응용 프로그램으로, 운영 서비스가 요청된 시각이나 이를 요청한 프로세스의 이름 등 시스템 동작에 대한 정보를 기록한다.
- **칩셋** chipset 컨트롤러와 보조 프로세서, 버스, 기타 하드웨어에 특정한 요소들의 집합으로, 시스템의 하드웨어 능력을 결정한다.
- **카운팅 세마포어** counting semaphore 1 이상의 값을 가질 수 있는 세마포어로, 대개 동일한 자원들을 담고 있는 풀에서 자원을 할당할 때 사용된다.
- **캐시 라인** cache line 캐시 안에 있는 엔트리
- **캐시 메모리** cache memory 소량, 고가의 고속 메모리로, 프로그램과 데이터 사본을 보유해 메모리 접근 시간을 줄여준다.
- **캐시 실패** cache miss 캐시에 없는 데이터에 대한 요청
- **캐시 적중** cache hit 캐시에 있는 데이터에 대한 요청
- **캐시콜드 프로세스** cache-cold process 자신이 디스패치될 프로세서의 캐시에 명령어와 데이터를 거의 두지 않은 프로세스
- **캐시핫 프로세스** cache-hot process 대부분의 데이터와 명령어를 자신이 디스패치될 프로세서의 캐시에 두는 프로세스
- **캡슐화 보안 페이로드** ESP, Encapsulating Security Payload **(IPSec)** 메시지 데이터를 대칭키 암호화를 사용해 암호화함으로써 IP 패킷이 공중 통신 회선을 통해 전송되는 동안 도청되지 않도록 하는 것
- **커널** kernel 운영체제의 핵심 구성 요소를 담고 있는 소프트웨어
- **커널 모드** kernel mode 프로세서의 실행 모드로, 프로세스들이 특권 명령어를 실행할 수 있게 해준다.
- **커널 세마포어** kernel semaphore **(리눅스)** 상호 배제를 제공하기 위해 커널에서 구현한 세마포어
- **커널 수준 스레드** kernel-level thread 운영체제에서 생성한 스레드로, 흔히 커널 스레드라고도 한다.
- **커널 스레드** kernel thread **(리눅스)** 커널 코드를 실행하는 스레드
- **커널 제어 경로** kernel control path **(리눅스)** 커널 자료 구조에 대한 상호 배타적인 접근을 요청하는 연산을 수행할 수도 있는 커널 실행 문맥
- **커밋한 트랜잭션** committed transaction 성공적으로 완료한 트랜잭션
- **커베로스** Kerberos 자유롭게 사용할 수 있는 오픈 소스 인증 및 접근 제어 프로토콜로, MIT에서 개발했으며 내부의 보안 공격으로부터 보호해준다. 커베로스는 비밀키 암호 기법을 사용해 네트워크의 사용자를 인증하고 네트워크 통신의 프라이버시와 무결성을 유지한다.
- **컨트롤러** controller 장치들이 버스에 접근하는 것을 관리하는 하드웨어 구성 요소
- **컴파일** compile 고급 언어로 작성한 소스 코드를 어셈블리 언어나 기계어로 변환하는 일

- **컴파일러** compiler 고급 언어로 작성한 소스 코드를 기계어로 변환하는 응용 프로그램
- **코드 동결** code freeze **(리눅스)** 알려진 버그를 수정하기 위한 코드가 아니라면 더는 커널에 새로운 코드를 추가할 수 없게 하는 시점
- **코드 생성기** code generator 컴파일러의 한 부분으로, 고급 언어로 작성한 소스에서 오브젝트 코드를 생성하는 역할을 담당한다.
- **코볼** COBOL, COmmon Business Oriented Language 1950년대에 개발된 절차적 프로그래밍 언어로, 대용량 데이터를 조작하는 비즈니스 소프트웨어를 작성하는 데 사용되었다.
- **코어 파일** core file **(리눅스)** 프로세스의 실행 상태를 담는 파일로, 주로 프로세스가 치명적인 예외를 만난 후 디버깅하는 목적으로 사용한다.
- **쿼드 펌핑** quad pumping 클록 사이클당 메모리 전송을 네 번 수행해 프로세서 성능을 향상하는 기술
- **퀀텀** quantum 프로세스가 프로세서를 빼앗기기 전에 실행할 수 있는 시간 단위. 퀀텀을 지정해 프로세스가 프로세서를 독점하는 일을 방지할 수 있다.
- **크기** size **(파일)** 파일에 저장된 정보의 양
- **크래커** cracker 악의를 가지고 시스템을 침투해 서비스를 못하도록 하거나 데이터를 훔치는 데 관심을 가진 사람
- **큰 단위 스트립** coarse-grained strip **(RAID)** 평균적인 파일들이 적은 수의 스트립에 저장되게 해주는 스트립 크기. 이 경우, 몇몇 요청을 디스크 배열의 한 부분을 통해 서비스할 수 있고, 따라서 다수의 요청을 동시에 서비스할 수 있는 확률이 높아진다. 요청들이 작으면 한 번에 한 디스크에 의해 서비스될 수 있고, 이는 평균 전송률을 줄여준다.
- **클라이언트** client 또 다른 프로세스(서버)에 서비스를 요청하는 프로세스. 클라이언트 프로세스가 작동하는 컴퓨터 역시 클라이언트라고 한다.
- **클라이언트 스텁** client stub 클라이언트 측에 있는 스텁으로, 외부로 나가는 데이터를 전송 준비시키고, 들어오는 데이터를 변환해 올바르게 해석할 수 있게 한다.
- **클래스** class 객체의 타입으로, 한 객체의 메소드와 속성을 정의한다.
- **클록 페이지 교체 전략** clock page-replacement strategy 2차 기회 페이지 교체 전략의 변형으로, 페이지들을 선형 리스트 대신 원형 리스트에 배치한다. 리스트 포인터들은 시계 바늘과 같이 원형 리스트 주위를 이동하고, 참조 비트가 'off'인 포인터 근처의 페이지를 교체한다.
- **클록틱** clocktick '사이클' 참고
- **키** key 데이터를 암호화할 때, 암호의 입력. 키들은 디지트 시퀀스로 나타난다.
- **키 분배 센터** KDC, Key Distribution Center 네트워크의 모든 사용자 간에 다른 비밀키를 공유하는 중앙 집중 기관
- **키 생성** key generation 암호화키를 생성하는 일
- **키 합의 프로토콜** key agreement protocol 안전하지 않은 매체를 통한 두 당사자 간의 키 교환을 관장하는 규칙들

- **타깃 컴퓨터** target computer **(프로세스 이동)** 프로세스가 이동하는 대상 컴퓨터
- **타이밍 제약** timing constraint 프로세스 혹은 프로세스 명령어의 부분 집합이 완료해야 하는 시간
- **타임 슬라이스** time slice '퀀텀' 참고
- **타임 슬라이싱** time slicing 각 프로세스가 선점되기 전에 많아도 한 퀀텀 기간 동안 실행되게 하는 스케줄링
- **타임스탬프** time stamp 메시지가 전송되는 지역 시각 기록
- **타임스탬핑** timestamping **(부인 방지)** 디지털 문서에 시간과 날짜를 결합하는 기술로, 부인 방지 문제를 해결하는 데 유용하게 사용된다.
- **타임스탬핑 대리자** time-stamping agency 디지털로 서명된 문서에 디지털 방식으로 타임스탬프를 찍는 기관
- **탄생 상태** born state 새로 생성된 스레드가 생명 주기를 처음 시작할 때의 상태
- **탐색 시간** seek time 읽기/쓰기 헤드가 현재 실린더에서 요청된 데이터 레코드를 갖고 있는 실린더로 옮기는 데 걸리는 시간
- **탐색 연산** seek operation 디스크 헤드를 다른 실린더로 옮기는 연산
- **탐색 최적화** seek optimization 읽기/쓰기 헤드 근처에 있는 실린더로의 요청들을 서비스함으로써 탐색 시간을 줄이는 디스크 스케줄링 기술
- **테스트 후 설정** test-and-set 하드웨어에서 구현한 명령어로, 변수 값을 확인한 후 true로 설정하는 일을 원자적으로 수행한다. 이 명령어는 스레드가 '읽기-수정-쓰기' 연산을 수행하는 동안 선점될 가능성을 제거해 상호 배제 구현을 간단하게 만든다.
- **테이블 단편화** table fragmentation 블록 맵핑 테이블로 소비되는 낭비 메모리. 블록이 작으면 시스템의 블록 수를 늘려 테이블 단편화를 촉진하는 경향이 있다.
- **텍스트 영역** text region 프로세스의 주소 공간 중 프로세서가 실행하는 명령어를 담은 영역
- **토큰** token 프로그램에 있는 문자로, 어휘 분석기로 분리한다. 토큰은 대체로 키워드, 식별자, 수학 연산자, 구두점 등을 나타낸다.
- **토큰** token **(토큰 기반 인증)** 인증에 대한 고유한 식별자
- **토큰 기반 인증** token-based authentication 각 세션마다 고유한 토큰을 발행하는 인증 기술로, 사용자가 특정 응용 프로그램에 접근할 수 있게 해준다.
- **통신 교착 상태** communication deadlock 분산 교착 상태의 한 형태로, 통신 신호를 순환 대기하는 상태다.
- **통지** notify 스레드 연산 중 하나로, 대상 스레드를 대기 상태에서 준비 상태가 되게 한다.
- **통합 장치 모델** unified device model **(리눅스)** 장치들을 디바이스 드라이버, 장치 클래스, 시스템 버스에 연관시키는 내부 장치 표현. 통합 장치 모델은 전력 관리와 핫 스왑 가능한 장치 관리를 간략하게 해준다.
- **투명성** transparency 분산 시스템에서 분산 양상을 사용자에게 감추는 것

- **튜플** tuple 테이블의 특정 요소
- **트랙** track 플래터에 있는 데이터의 원형 영역. 순차적인 파일 데이터들은 대체로 한 트랙에 연속되게 놓여 탐색 연산을 줄이고 접근 시간을 향상한다.
- **트랜잭션** transaction 더는 분할할 수 없는 원자적이고 상호 배타적인 연산으로, 전체가 완료되거나, 그렇지 않으면 롤백된다. 데이터베이스 엔트리에 대한 수정은 성능을 높이고 교착 상태 복구라는 고비용 작업을 피하기 위해 트랜잭션으로 이루어지는 것이 보통이다.
- **트랜잭션 대기 그래프** TWFG, Transaction Wait-For Graph 프로세스를 노드로, 의존 관계를 방향선으로 나타낸 그래프. 분산 교착 상태 탐지 알고리즘에서 사용된다.
- **트랜잭션 롤백** roll back a transaction 시스템을 트랜잭션 수행 이전 상태로 되돌리는 일
- **트랜잭션 투명성** transaction transparency 시스템이 일련의 자원들의 협력 관계를 감춤으로써 일관성을 유지하게 하는 속성
- **트랜지스터** transistor 전류가 흐르게 하거나 막음으로써 프로세서가 비트 단위로 연산할 수 있게 하는 소형 스위치
- **트랩** trap IA-32 명세에서는 오버플로우(레지스터에 저장된 값이 레지스터 용량을 초과할 때 발생)와 같은 오류 때문에 생성되는 예외를 말한다. 프로그램 제어가 코드의 중단점에 도달할 때도 트랩이 생성된다.
- **트레이스** trace 버스의 일부를 형성하는 가는 전도체 선
- **트로이 목마** Trojan horse 신뢰성 있는 프로그램 안에 숨어 있거나 합법적인 프로그램의 식별자나 기능을 가장하지만, 프로그램이 실행될 때 컴퓨터나 네트워크에 해를 입히는 악의적인 프로그램
- **특권 명령어** privileged instruction 커널 모드로만 실행할 수 있는 명령어. 특권 명령어는 보호되는 하드웨어와 소프트웨어 자원에 접근하는 연산을 수행한다(예를 들면, 프로세스 사이에서 프로세서를 바꾸거나 하드 디스크에 명령하는 연산).
- **티켓 부여 서비스** TGS, Ticket Granting Service **(커베로스)** 클라이언트가 특정 네트워크 서비스에 접근할 수 있는 권한을 인증해주는 서버
- **티켓 부여 티켓** Ticket-Granting Ticket **(커베로스)** 커베로스 인증 서버에서 반환된 티켓으로, 인증 서버와 공유되는 클라이언트의 비밀키로 암호화된 것이다. 클라이언트는 복호화된 TGT를 티켓 부여 서비스로 보내 서비스 티켓을 요청한다.
- **파괴** destroy **(파일)** 파일 시스템에서 파일을 삭제하는 연산
- **파서** parser 컴파일러의 일부로, 어휘 분석기로부터 토큰 스트림을 받고, 이 토큰들을 그룹화해 중간 코드 생성기에서 처리할 수 있게 한다.
- **파스칼** Pascal 구조적 프로그래밍 언어로, 1971년에 니클라우스 워스가 개발했다. 파스칼은 프로그래밍 입문 과정을 강의하는 데 많이 사용되었다.
- **파이버** fiber **(윈도우 XP)** 윈도우 XP의 실행 단위로, 스레드가 생성항고, 스케줄링한다. 파이버는 자신의 스레드를 스케줄링하는 응용 프로그램의 이식성을 높여준다.

- **파이버 지역 저장소** FLS, Fiber Local Storage **(윈도우 XP)** 프로세스의 주소 공간 중에서 특정 파이버가 자기만 접근할 수 있는 정보를 저장하는 영역
- **파이프** pipe 프로세스 간에 정보를 전송하기 위해 파이프를 선입선출 방식 버퍼처럼 사용하는 IPC 메커니즘
- **파이프 버퍼** pipe buffer **(리눅스)** 파이프에 쓴 데이터를 버퍼링하는 데 사용하는 데이터 페이지
- **파일** file 이름을 붙인 데이터 집합으로 열기 open, 닫기 close, 생성 create, 파괴 destroy, 복사 copy, 이름 변경 rename, 리스트 list 등의 연산을 통해 조작하는 단위다. 파일에 있는 개별 데이터 항목은 읽기 read, 쓰기 write, 갱신 update, 삽입 insert, 삭제 delete 등의 연산으로 조작할 수 있다. 파일 특성에는 접근성, 유형, 휘발성, 활성도 등이 있다. 파일은 레코드 하나 이상으로 구성된다.
- **파일 관리** file management 파일 시스템 기능 중 하나로, 파일을 저장, 조회, 공유하고 보안을 지키는 메커니즘을 제공한다.
- **파일 구성** file organization 파일의 레코드들이 2차 저장소에 정렬되는 방식(예를 들면, 순차 방식, 직접, 인덱스를 사용한 순차 방식, 파티션 사용 등)
- **파일 기술자** file descriptor 음수가 아닌 정수로, 열린 파일 테이블의 인덱스로 사용된다. 프로세스는 경로명 대신 파일 기술자를 사용해 디렉토리 탐색을 위한 오버헤드 없이 파일 데이터에 접근할 수 있다.
- **파일 무결성 메커니즘** file integrity mechanism 파일에 있는 정보가 손상되지 않음을 보장하는 기술. 파일의 무결성이 보장될 때, 파일은 의도된 정보만 포함한다.
- **파일 서버** file server 원격 프로세스들이 파일에 접근할 수 있게 해주는 전용 시스템
- **파일 속성** file attribute **(리눅스)** 접근 제어 정보를 구현하는 파일 메타데이터로, 파일이 추가 전용인지, 변경 불가인지, 표준 리눅스 파일 권한으로 명시할 수 없는지 등의 정보를 나타낸다.
- **파일 시스템** file system 운영체제 구성 요소로, 파일을 구성하고 데이터에 대한 접근을 관리한다. 파일 시스템은 (경로명을 사용해) 파일을 논리적으로 구성하고, (메타데이터를 사용해) 물리적으로 구성하는 일을 담당한다. 또한 저장 장치의 여유 공간을 관리하고 보안 정책을 강화하고 데이터의 무결성을 유지하기도 한다.
- **파일 시스템 관리자** file system manager 운영체제 구성 요소로, 저장 장치에 있는 데이터를 이름 있는 집합으로 구성하고, 해당 장치에 접근할 수 있게 인터페이스를 제공하는 역할을 한다.
- **파일 시스템 식별자** file system identifier 저장 장치가 사용하고 있는 파일 시스템을 고유하게 식별할 수 있는 값
- **파일 제어 블록** file control block 시스템이 파일을 관리하는 데 필요한 정보(예를 들어, 접근 통제 정보 등)를 담고 있는 메타데이터
- **파일 할당 테이블** FAT, File Allocation Table 마이크로소프트의 FAT 파일 시스템에서 파일 데이터 블록을 가리키는 포인터들을 저장하는 테이블
- **파일 허가** file permission 사용자가 특정 파일을 읽기, 쓰기, 실행할 수 있는 권한을 결정하는 구조
- **파티션** partition **(파일 시스템)** 파일이 경계를 걸칠 수 없는 디스크의 영역. 파티션들은 디스크 단편화를 줄일 수 있다.

- **파티션** partition **(실제 메모리 구성)** 고정/가변 파티션 멀티프로그래밍에서 프로세스에 할당된 메인 메모리의 부분. 프로그램은 파티션에 배치되어 운영체제가 사용자 프로세스로부터 자신을 보호하고 각 프로세스를 다른 프로세스들로부터 보호할 수 있게 한다.
- **파티션된 파일** partitioned file 순차적인 하위 파일들로 구성된 파일
- **패리티** parity 데이터 전송에서 짝수의 오류를 탐지하는 기술. 패리티 정보는 데이터가 짝수(혹은 홀수) 개의 1(또는 0)을 포함하고 있는지 결정함으로써 생성할 수 있다. 이 패리티 정보는 전송 후의 정보고, 전송 전에 생성된 값과 비교된다. 해밍 코드나 XOR 코드와 같은 오류 정정 코드는 오류를 탐지하고 정정하기 위해 패리티 문자열을 사용한다. 패리티는 미러링된 시스템보다 낮은 저장소 오버헤드로 RAID 시스템이 장애 내구성을 제공하게 해준다.
- **패리티 로깅** parity logging **(RAID)** 패리티를 사용하는 RAID 시스템에서 쓰기 성능을 향상하는 기술로, 시스템이 바쁜 동안에는 패리티 디스크에 쓰는 것을 연기한다. 패리티 로깅은 휘발성 메모리에 정보를 저장하기 때문에, 패리티 정보는 시스템 전원이 나가면 없어진다.
- **패스워드 보호** password protection 자원이나 시스템에 대한 접근 권한을 얻기 위해 사용자가 나타내는 이름과 이에 대응하는 패스워드에 근거해 인증하는 기술
- **패스워드 솔팅** password salting 암호화 전에 패스워드의 다양한 위치에 문자를 삽입해 넣는 기술로, 브루트포스 공격에 대한 취약점을 줄여준다.
- **패스워드 에이징** password aging 사용자가 주기적으로 패스워드를 갱신하게 해 보안 수준을 높이는 기술
- **패키지** package **(리눅스)** 응용 프로그램이나 서비스를 담은 배포 부분. 사용자는 패키지를 추가, 삭제해 리눅스 시스템을 커스터마이즈할 수 있다.
- **패킷 필터링 방화벽** packet-filtering firewall LAN 밖에서 보낸 모든 데이터를 검사하는 하드웨어나 소프트웨어로, 미리 정의된 규칙에 따라 특정 데이터 패킷을 거절한다. 예를 들어, 지역 네트워크 주소를 가진 패킷이나 특정 주소 혹은 포트에서 보낸 패킷을 거부한다.
- **펌웨어** firmware 기계어 명령어를 구현하는 데 필수적인 간단하고 기초적인 명령어들을 지정하는 마이크로코드
- **페이로드** payload 특정 조건이 충족되면 실행되는 논리폭탄 내부의 코드
- **페이지** page 프로세스의 가상 주소 공간에서 한 단위로 관리되는 고정 크기의 연속된 주소 집합. 페이지는 프로세스의 데이터와 명령어 일부를 담으며 메인 메모리의 여유 페이지 프레임에 놓일 수 있다.
- **페이지 교체 전략** page-replacement strategy 들어오는 페이지를 위해 메모리 공간을 마련하기 위해 어떤 페이지를 내보낼 것인지 결정하는 전략. 많은 페이지 교체 전략들은 미래의 페이지 사용을 예측해 최적의 성능을 내려고 노력한다.
- **페이지 디렉토리 엔트리** page directory entry **(IA-32 인텔 아키텍처)** 페이지 디렉토리에 있는 개체로, 페이지 테이블 엔트리를 저장하는 페이지 테이블의 기본 주소에 맵핑된다.
- **페이지 맵 테이블** page map table '페이지 테이블' 참고

- **페이지 언맵핑** unmap a page **(리눅스)** 대응하는 페이지가 더는 메모리에 없음을 나타내기 위해 페이지 테이블 엔트리를 갱신하는 일
- **페이지 전역 디렉토리** page global directory 2계층 다수준 페이지 테이블에서 페이지 전역 디렉토리는 프로세스의 페이지 테이블의 일부에 대한 포인터 테이블이다. 페이지 전역 디렉토리들은 다수준 페이지 테이블 계층의 최상위에 있다.
- **페이지 전역 디렉토리** page global directory **(리눅스)** 가상 메모리 구조로 2차 수준 페이지 맵핑 테이블의 주소를 저장한다.
- **페이지 중간 디렉토리** page middle directory **(리눅스)** 가상 메모리 구조로 3차 수준 페이지 맵핑 테이블의 주소들을 저장한다. 페이지 테이블이라고도 한다.
- **페이지 캐시** page cache **(리눅스)** 디스크의 데이터 페이지들을 저장하는 캐시. 프로세스가 디스크로부터 데이터를 요청할 때, 커널은 먼저 해당 페이지가 캐시에 있는지를 확인해 값비싼 디스크 입출력 연산을 줄인다.
- **페이지 테이블** page table 페이지 번호를 페이지 프레임에 맵핑하는 엔트리를 저장하는 테이블. 페이지 테이블은 프로세스의 각 가상 페이지에 대응하는 엔트리를 담고 있다.
- **페이지 테이블 시작점 레지스터** page table origin register 메인 메모리에 있는 프로세스의 페이지 테이블 위치를 저장하는 레지스터. 이 정보를 고속 레지스터에 보유함으로써 가상-물리 주소 변환을 빠르게 수행할 수 있다.
- **페이지 테이블 엔트리** PTE, Page Table Entry 페이지 테이블에 있는 엔트리로, 가상 페이지 번호를 페이지 프레임 번호에 맵핑한다. 페이지 테이블 엔트리는 페이지에 관한 다른 정보(예를 들면, 페이지 접근 방법, 페이지가 메모리에 상주하는지 여부) 등을 저장한다.
- **페이지 폴트** page fault 프로세스가 메모리에 없는 페이지를 참조하려고 시도할 때 발생하는 오류로, 이 경우 운영체제는 디스크에서 필요한 페이지를 로드해야 한다.
- **페이지 폴트 빈도 페이지 교체** PFF(Page-Fault-Frequency) page replacement 프로세스가 페이지 폴트를 마주치는 빈도에 따라 프로세스의 상주 페이지 집합을 조절하는 알고리즘. 만일 프로세스가 큰 작업 집합으로 변해가면, 부재 오류가 빈번히 발생할 것이고, PFF는 페이지 프레임을 더 많이 할당해준다. 프로세스가 새로운 작업 집합을 메모리에 다 모으면, 페이지 폴트가 안정화될 것이고, PFF는 페이지들을 메모리에 유지하거나 혹은 감소시킬 것이다. 적절하고 효율적인 PFF의 핵심은 그 기준 임계치를 적절하게 유지하는 것이다.
- **페이지 프레임** page frame 가상 페이지를 저장할 수 있는 메인 메모리 블록. 단일 페이지 크기를 갖는 시스템에서는 어떤 페이지든 여유 페이지 프레임에 놓일 수 있다.
- **페이지 플러싱** flushing a page 메인 메모리의 수정된 페이지를 2차 저장소로 복사해, 내용을 손실하지 않고 다른 페이지가 그 프레임에 배치될 수 있도록 하는 것. 페이지 플러싱이 발생할 때, 페이지의 수정 비트를 초기화해, 페이지를 덮어쓸 수 있음을 운영체제가 빨리 알게 해 페이지 대기 시간을 줄인다.
- **페이징** paging 가상 메모리 구성 기술로, 주소 공간을 고정 크기의 연속된 주소 블록으로 나눈다. 프로세스의 가상 주소 공간에 적용되면, 이 블록들을 페이지라고 한다. 페이지는 프로세스 데이터와 명령어들을 저장한다. 메인 메모리에 적용하면, 블록은 페이지 프레임이라고 한다. 페이지는 2차 저장소에 저장되었다가 페이지 프레임

에 로드된다. 페이징은 메모리 배치 결정을 간단히 해주고 외부 단편화를 야기하지 않는다(단일한 페이지 크기를 가진 시스템에서). 페이징은 내부 단편화를 야기하지 않는다.

- **페치 전략** fetch strategy 다음 프로그램이나 데이터 부분을 2차 저장소로부터 메인 메모리에 로드할 시점을 결정하는 방법
- **평균 고장 시간** MTTF(Mean-Time-To-Failure) **(RAID)** 단일 디스크가 고장 날 때까지의 평균 시간
- **평균 반응 시간** mean response time **(디스크 스케줄링)** 시스템이 디스크 요청이 서비스될 때까지 대기하는 평균 시간
- **평면 디렉토리 구조** flat directory structure 한 디렉토리만 포함하는 파일 시스템 구성
- **평문** plaintext 암호화되지 않은 데이터
- **포트** port 두 장치를 연결하는 버스
- **포트란** Fortran 절차적 프로그래밍 언어로, 1950년대 중반에 복잡한 수학 계산이 필요한 과학적 연구를 위해 IBM에서 개발했다.
- **포화 임계치** saturation threshold 자원 활용도 수준으로, 이 임계치를 초과하면 해당 자원에 대한 접근을 거부한다. 교착 상태를 방지하기 위해 설계되었지만, 처리량도 감소시키는 경향이 있다.
- **폴링** polling 각 장치를 주기적으로 확인해 하드웨어 상태를 알아내는 기술. 인터럽트 대신 폴링을 사용할 수 있지만, 오버헤드가 증가해 성능을 떨어뜨릴 수 있다.
- **폴트** fault 인텔 IA-32 명세에서 폴트는 0으로 나누려 하거나 메모리에 불법적으로 접근하는 등의 오류 때문에 발생한다. 어떤 폴트는 운영체제가 적절한 예외 처리기를 수행해 정정할 수도 있다.
- **폴트 간 시간** interfault time 프로세스의 페이지 폴트 사이의 시간. 이 수치는 페이지 폴트 빈도를 이용한 페이지 교체 전략에서 사용되어 언제 프로그램의 페이지 프레임 할당을 증가 혹은 감소시킬지 결정할 수 있게 한다.
- **표준화된 인터페이스** standardized interface 각 클라이언트/서버 쌍이 서로가 이해할 수 있고 이를 사용해 통신할 수 있도록 해주는 단일한 공통 인터페이스
- **프라이머리 키** primary key 관계형 데이터베이스에서 튜플을 고유하게 식별하는 값을 갖는 속성
- **프라이버시** privacy **(보안 트랜잭션)** 성공적이고 안전한 트랜잭션을 위한 다섯 가지 기본 요소 중 하나로, 인터넷을 통해 전송된 정보를 제3자가 캡처하거나 사용자가 모르는 곳으로 전송하지 않았음을 보장하는 문제를 다룬다.
- **프라이버시 설정 플랫폼** P3P, Platform for Privacy Preferences 사용자가 사이트에서 수집하는 개인 정보를 제어할 수 있게 해 싱글 사인 온과 다른 응용 프로그램에 전송된 정보의 프라이버시를 보호하는 플랫폼
- **프로그래밍된 입출력** PIO, Programmed I/O 인터럽트를 지원하지 않는 장치들을 위한 입출력 구현. 이러한 경우에는 프로세서가 메모리로부터 전송되는 각 워드를 감독해야 한다.
- **프로그램 동작의 작업 집합 이론** working set theory of program behavior 데닝의 이론으로, 한 프로그램이 효율적으로 실행되려면 시스템이 프로그램의 선호하는 페이지 부분 집합(즉, 작업 집합)을 메모리에 유지해야 한다는 것이다. 작업 집합 윈도우 w가 주어졌을 때, 프로세스의 작업 집합 $W(t, w)$는 $t-w$에서부터 t까지의 프로세스 시간

동안 참조한 페이지들을 가리킨다. 윈도우 크기 w를 지정하는 것은 작업 집합 메모리 관리를 구현하는 데 중요한 영향을 미친다.

- **프로그램 카운터** program counter 프로세서가 실행 중인 프로세스를 위해 실행할 명령어를 가리키는 포인터. 프로세서가 한 명령어를 수행하고 나면, 프로그램 카운터는 프로세서가 실행할 다음 명령어를 가리키도록 이동한다.
- **프로세서** processor 기계어 명령어를 실행하고 메인 메모리와 같은 시스템 자원을 보호하는 하드웨어 구성 요소
- **프로세서 스케줄링 규칙** processor scheduling discipline '프로세서 스케줄링 정책' 참조
- **프로세서 스케줄링 정책** processor scheduling policy 시스템이 프로세스들에 언제, 얼마 동안 프로세서를 할당할 것인지 결정하는 데 사용하는 전략
- **프로세서 중심** processor-bound 실행 시 자신의 퀀텀을 소비하는 프로세스(또는 작업). 이러한 프로세스(또는 작업)는 입출력 요청이 적고, 계산의 양이 많다는 특징이 있다.
- **프로세서 풀** processor pool 아메바 시스템의 구성 요소로, 프로세서 집합을 보유하고 있고, 각 프로세서 풀은 자신의 메모리와 이더넷 연결을 가지고 있다.
- **프로세스** process 실행 중인 프로그램을 나타내는 개체
- **프로세스 간 통신 관리자** interprocess communication manager 운영체제 구성 요소로, 프로세스 간 통신을 관리한다.
- **프로세스 기술자** process descriptor '프로세스 제어 블록' 참고
- **프로세스 복제** process cloning 원격 머신에 프로세스의 사본을 생성하는 것
- **프로세스 상태** process state '실행', '준비', '블록' 상태 같은 한 프로세스의 상태
- **프로세스 생성** spawning a process 부모 프로세스가 자식 프로세스를 생성하는 일
- **프로세스 스케줄러** process scheduler 운영체제 구성 요소로, 어떤 프로세스가 프로세서에 접근할 것인지, 프로세서를 얼마나 사용할지를 결정한다.
- **프로세스 식별 번호** PID(Process IDentification) number 한 프로세스를 유일하게 식별할 수 있는 값
- **프로세스 식별자** PID, Process IDentifier 프로세스를 고유하게 식별하는 정수 값
- **프로세스 우선순위** priority of a process 다른 프로세스 대비 해당 프로세스의 중요성과 긴급성
- **프로세스 우선순위** process priority 프로세스의 중요도를 다른 프로세스들에 비해 결정해주는 값. 프로세스 스케줄러는 이 값을 사용해 프로세스의 실행 순서를 결정한다.
- **프로세스 제어 블록** PCB, Process Control Block 한 프로세스의 특징적인 정보(예를 들면, PID, 주소 공간, 상태 등)를 저장하는 자료 구조. 프로세스 기술자라고도 한다.
- **프로세스 테이블** process table 알려진 프로세스들의 테이블. 세그먼테이션/페이징 시스템에서는 프로세스 테이블의 엔트리들이 프로세스의 가상 주소 공간을 가리킨다.
- **프로젝션** projection **(데이터베이스)** 속성의 하위 집합을 생성하는 연산

- **프로퍼티** property **(객체)** 객체의 일부분으로, 객체에 관한 데이터를 저장한다.
- **프록시** proxy DCOM에서 클라이언트 측 스팁으로, 메시지를 마셜링, 언마셜링하는 것을 담당한다.
- **프론트사이드 버스** FSB, FrontSide Bus 프로세서와 메인 메모리 또는 기타 주변 장치 사이를 연결하는 버스
- **프리페이징** prepaging '예측 페이징' 참고
- **프리페칭** prefetching '예측 페이징' 참고
- **플래터** platter 마그네틱 디스크 매체로 표면에 비트들을 저장한다.
- **플러그 가능한 인증 모듈** PAM, Pluggable Authentication Module **(리눅스)** 리눅스 시스템에서 인증을 강화하기 위해 실행 시에 설치될 수 있는 모듈
- **플러그 앤 플레이** plug-and-play 드라이버 설치와 운영체제의 하드웨어 구성을 쉽게 해주는 기술
- **필드** field 데이터 계층에서 문자들(예를 들면, 한 사람의 이름, 주소, 전화번호 등)의 그룹
- **하드 디스크 드라이브** hard disk drive 자성을 가진 회전식 2차 저장 장치로, 데이터를 영구 저장할 수 있고, 데이터에 대한 임의적 접근이 가능하다.
- **하드 링크** hard link 저장 장치에 있는 파일의 위치를 지정하는 디렉토리 엔트리
- **하드 실시간 스케줄링** hard real-time scheduling 프로세스들이 데드라인을 지킬 수 있도록 보장하는 스케줄링 정책
- **하벤더의 선형 순서** Havender's linear ordering '선형 순서' 참고
- **핫 스왑 가능한 장치** hot swappable device 컴퓨터가 실행되는 동안에 추가 또는 삭제될 수 있는 장치
- **핫 스팟** hot spot 자주 요청되는 데이터를 갖고 있는 디스크 실린더. 몇몇 디스크 암 예측 기술은 디스크 헤드를 핫 스팟으로 옮겨 평균 탐색 시간을 줄이기도 한다.
- **핫 스페어 디스크** hot spare disk **(RAID)** RAID 시스템에 있는 디스크로, 디스크 하나가 고장 날 때 사용된다. 시스템이 고장 난 디스크의 데이터를 재생성하고 나면, 핫 스페어 디스크는 고장 난 디스크를 대체한다.
- **해밍 오류 정정 코드** Hamming ECC(Error-Correcting Code) 패리티 비트를 생성해 시스템이 데이터 전송 시의 오류를 탐지하고 정정할 수 있게 해주는 기술
- **해시 값** hash value 해시 함수가 반환한 값으로, 해시 테이블의 위치에 대응한다.
- **해시 앵커 테이블** hash anchor table 역 페이지 테이블의 개체들을 가리키는 해시 테이블. 해시 앵커 테이블 크기가 늘어나면 충돌 수가 감소하고, 주소 변환 속도가 빨라진다. 그러나 테이블을 저장하기 위한 메모리 오버헤드는 증가한다.
- **해시 테이블** hash table 항목들을 해시 값에 따라 인덱싱하는 자료 구조. 해시 함수와 함께 사용해 데이터를 빠르게 저장하고 조회할 수 있다.
- **해시 함수** hash function 입력 수를 받고 출력 수(해시 값이라고 함)를 반환하는 함수. 해시 함수는 해시 테이블을 사용해 정보를 빨리 저장하고 조회할 수 있다.

- **해제 시간** teardown time 시스템 운영자와 운영체제가 작업이 완료된 후 시스템에서 해당 작업을 제거하는 데 걸리는 시간
- **해커** hacker 숙련된 프로그래머를 지칭하는 말로, 이들은 흔히 응용 프로그램의 기능을 작성하는 것만큼이나 개인적인 흥미를 위해 프로그래밍할 때가 많다. 이 용어는 흔히 '크랙커'라는 용어가 더 적절할 때 대신 사용되기도 한다.
- **행동** behavior **(객체)** '메소드' 참조
- **협력형 멀티태스킹** cooperative multitasking 프로세스 스케줄링 기술로, 프로세스는 자발적으로 제어를 반납할 때까지 프로세서에서 실행할 수 있다.
- **호스트 기반 침입 탐지** host-based intrusion detection 시스템과 응용 프로그램의 로그 파일들을 모니터링하는 침입 탐지 시스템
- **홀** hole **(메모리)** 가변 파티션 멀티프로그래밍 시스템에서 '사용되지 않은' 메모리 영역
- **홈 컴퓨터** home computer **(프로세스 이동)** 프로세스가 원래 있던 컴퓨터
- **확장 펌웨어 인터페이스** EFI, Extensible Firmware Interface 인텔에서 고안한 인터페이스로, 디바이스 드라이버를 지원하고 부팅 시 셸 인터페이스를 제공해 전통적인 바이오스를 개선했다.
- **확장성 있는 운영체제** extensible operating system 새로운 기능과 변화를 쉽게 수용할 수 있는 운영체제
- **환영 교착 상태** phantom deadlock 분산 컴퓨팅과 관련된 통신 지연에 의해 발생하는 것으로, 교착 상태 탐지 알고리즘 DDA은 실제로는 존재하지 않는 교착 상태를 탐지할 수도 있다.
- **활성 리스트** active list **(리눅스)** 현재 시기 epoch 동안에 적어도 한 번 프로세서를 제어하는 프로세스를 담은 스케줄러 구조
- **활성 상태** active state **(리눅스)** 현재 시기 동안에 프로세서에서 실행하려고 경쟁할 수 있는 작업 상태
- **활성 페이지** active page **(리눅스)** 메모리의 페이지 중 다음 번에 교체 페이지를 선택할 때 교체 대상이 되지 않는 페이지
- **활성도** activity **(파일)** 주어진 시간 안에 파일의 레코드를 참조하는 비율
- **회전 지연** rotational latency 디스크가 회전해 요청된 데이터 항목이 현재 위치에서 읽기/쓰기 헤드 근처에 오도록 하는 데 걸리는 시간
- **회전 최적화** rotational optimization 읽기/쓰기 헤드의 현재 실린더에 있는 가장 가까운 섹터로의 요청을 서비스함으로써 접근 시간을 줄이는 디스크 스케줄링 기술
- **효율적인 운영체제** efficient operating system 처리량이 많고 처리 시간이 짧은 운영체제
- **후면 쓰기 캐싱** write-back caching 버퍼에 쌓인 데이터를 주기적으로 디스크에 쓰는 기술. 이는 운영체제가 여러 입출력을 하나로 묶어 한 번의 요청으로 서비스되도록 해, 시스템 성능을 향상한다.

- **훅** hook 개발자들이 기존의 응용 프로그램 소스를 수정하지 않은 채 기능을 추가할 수 있게 하는 소프트웨어 기능. 응용 프로그램은 훅을 사용해 다른 응용 프로그램에서 정의할 수 있는 프로시저를 호출한다.
- **휘발성** volatility **(파일)** 파일에 대해 추가 삭제가 이루어지는 빈도
- **휘발성 저장소** volatile storage 전원이 나갈 때 데이터를 잃는 저장 매체
- **휴면 간격** sleep interval 휴면 상태에 진입하려는 스레드가 명시한 휴면 상태 시간
- **휴면 상태** sleeping state 지정된 시간이 만료될 때까지 실행될 수 없는 스레드 상태. 휴면 간격이 만료되고 나면 스레드는 준비 상태로 돌아온다.