

운영체제

운영체제 필수 지식을 체계적으로 학습해보겠습니다.





학습 목차

01

운영체제 기본 개념

OS의 역할과 시스템 콜의 핵심 이해

03

CPU 스케줄링

효율적인 프로세서 관리 알고리즘

05

메모리 관리

가상 메모리와 페이지 교체 전략

07

최신 기술

가상화와 컨테이너 기술 동향

02

프로세스 & 스레드

실행 단위의 특성과 상태 변화 과정

04

프로세스 동기화

임계구역과 교착상태 해결 방법

06

파일 시스템

저장장치 관리와 디스크 스케줄링

08

면접 핵심 질문

실무에서 자주 묻는 필수 문제들

1. 운영체제 기본 개념

운영체제의 핵심 역할



자원 관리

CPU, 메모리, I/O 장치를 효율적으로 분배하고 관리하여 시스템 성능을 최적화합니다.



추상화 제공

복잡한 하드웨어를 숨기고 응용프로그램에게 편리한 인터페이스를 제공합니다.



보안 및 보호

프로세스 간 메모리 격리와 접근 제어를 통해 시스템 안전성을 보장합니다.



서비스 제공

시스템 콜을 통해 파일 I/O, 프로세스 관리 등의 OS 서비스를 제공합니다.

시스템 콜의 이해

정의와 목적

사용자 프로그램이 운영체제 서비스를 안전하게 요청하는 인터페이스로, 커널 모드 전환을 통해 하드웨어에 접근합니다.

주요 종류

- 프로세스 제어: `fork()`, `exec()`, `wait()`, `exit()`
- 파일 조작: `open()`, `read()`, `write()`, `close()`
- 통신: `pipe()`, `shmget()`, `mmap()`

1

시스템 콜 호출

소프트웨어 인터럽트 발생

2

모드 전환

사용자 모드 → 커널 모드

3

서비스 처리

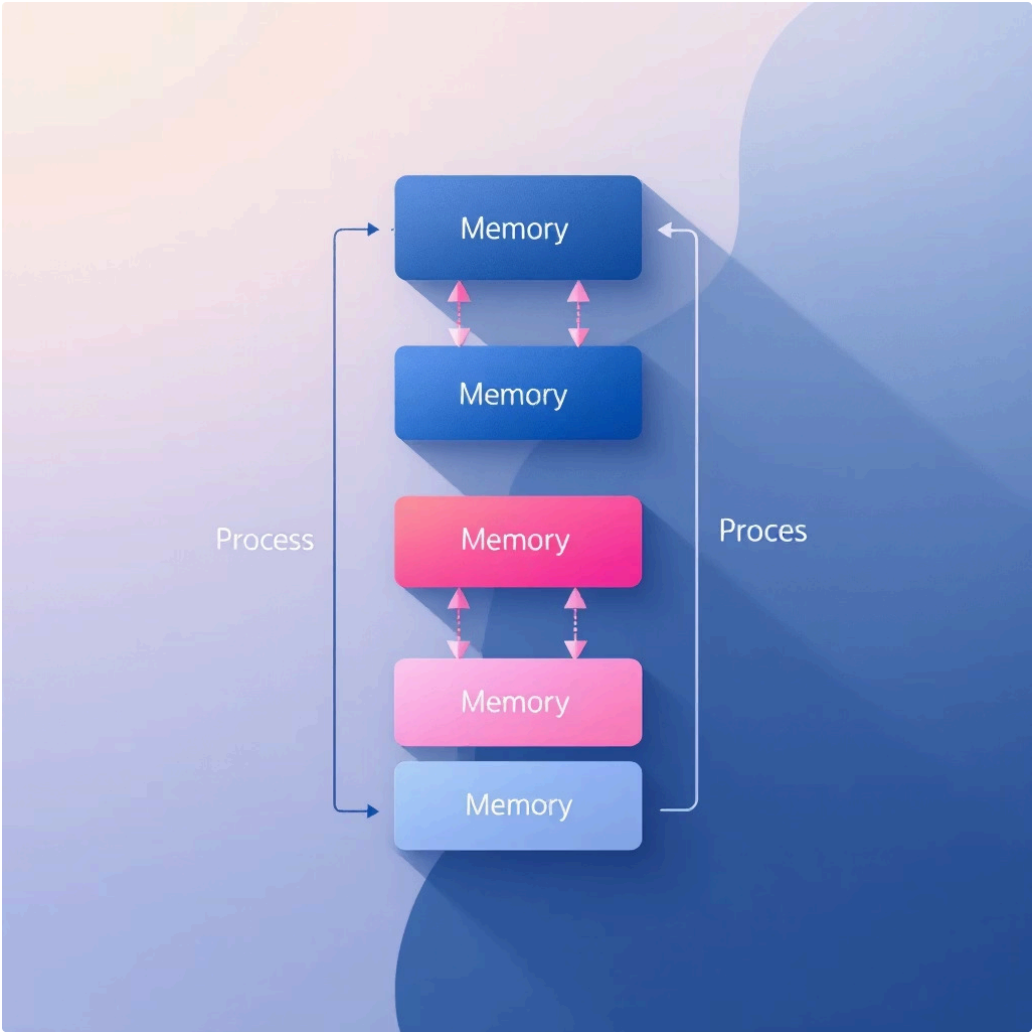
커널이 요청된 작업 수행

4

모드 복귀

커널 모드 → 사용자 모드

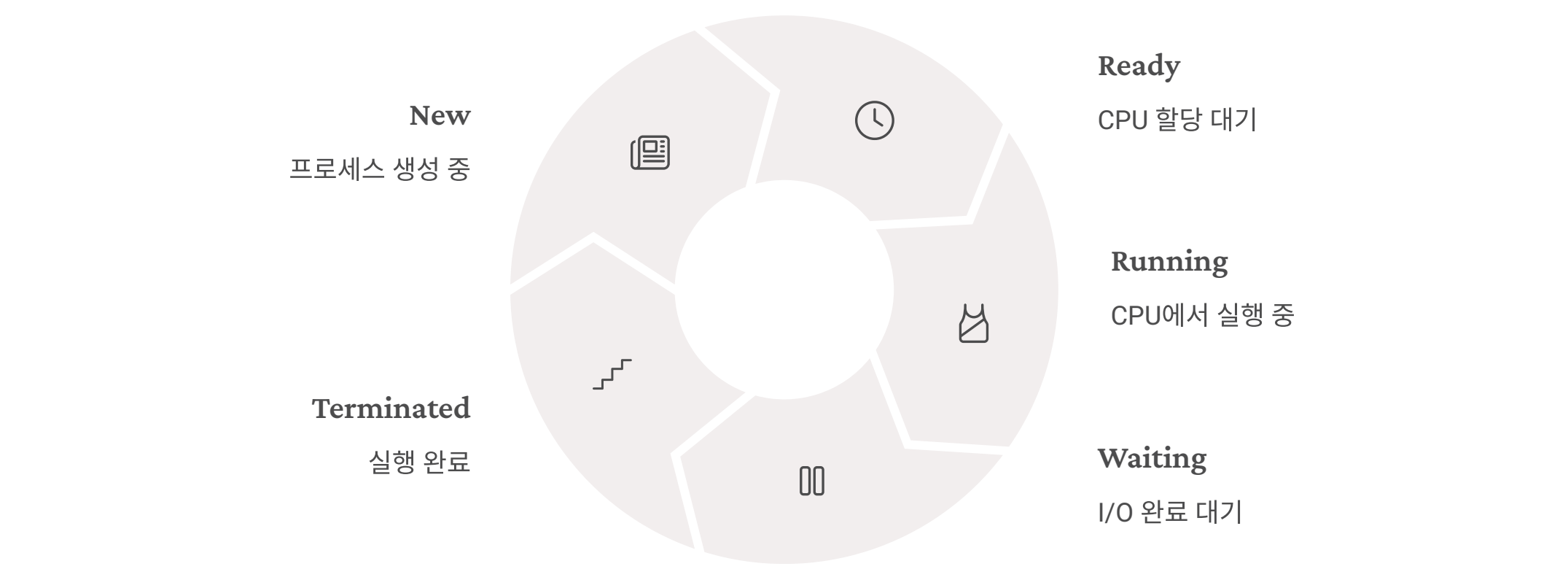
2. 프로세스 & 스레드



프로세스 정의

실행 중인 프로그램으로, 코드, 데이터, 힙, 스택으로 구성된 독립적인 메모리 공간을 갖습니다.

프로세스 상태 변화



프로세스 vs 스레드 비교

특성	프로세스	스레드
메모리	독립적 주소 공간	주소 공간 공유
생성 비용	높음 (PCB 생성)	낮음 (TCB만 생성)
통신	IPC 메커니즘 필요	메모리를 통한 직접 통신
보안	높음 (완전 격리)	낮음 (메모리 공유로 인한 위험)

컨텍스트 스위칭

정의

CPU 제어권이 한 프로세스에서 다른 프로세스로 넘어가는 과정으로, 현재 상태를 PCB에 저장하고 새 프로세스 상태를 복구합니다.

발생 상황


- 타이머 인터럽트로 인한 시간 할당량 초과
- I/O 요청으로 인한 프로세스 블록
- 높은 우선순위 프로세스의 도착


오버헤드


PCB 저장/복구 시간, 캐시 무효화, 메모리 관리 단위 재설정으로 인한 성능 저하가 발생합니다.

3. CPU 스케줄링

스케줄링 목표

- 

CPU 이용률 최대화
프로세서가 유휴 상태가 되지 않도록 하여 하드웨어 자원을 효율적으로 활용합니다.
- 

처리량 최대화
단위 시간당 완료되는 프로세스의 수를 늘려 전체 시스템 생산성을 향상시킵니다.
- 

응답 시간 최소화
사용자의 요청에 대한 첫 번째 응답까지의 시간을 줄여 사용자 경험을 개선합니다.

주요 스케줄링 알고리즘

- 1

FCFS (First Come First Served)
방식: 도착 순서대로 처리하는 비선점형 스케줄링
장점: 구현 간단, 공정함
단점: 콘보이 효과로 평균 대기시간 증가
- 2

SJF (Shortest Job First)
방식: 실행 시간이 가장 짧은 작업을 우선 처리
장점: 평균 대기시간 최소화 (이론적 최적)
단점: 기아 현상, 실행시간 예측 어려움
- 3

Round Robin
방식: 시간 할당량을 두고 순환하여 실행
장점: 공정성 보장, 응답시간 개선
단점: 문맥교환 오버헤드 발생
- 4

우선순위 스케줄링
방식: 우선순위가 높은 프로세스를 먼저 실행
문제: 낮은 우선순위 프로세스의 기아 현상
해결: 에이징 기법으로 우선순위 동적 증가
- 5

MLFQ (Multi-Level Feedback Queue)
방식: 여러 우선순위 큐 + 동적 우선순위 조정
장점: I/O 집약적 프로세스 우선 처리
실제 사용: 대부분의 현대 운영체제에서 채택

4. 프로세스 동기화

임계구역 문제

임계구역 정의

공유 자원에 접근하는 코드 영역으로, 동시에 여러 프로세스가 실행되면 데이터 불일치가 발생할 수 있습니다.

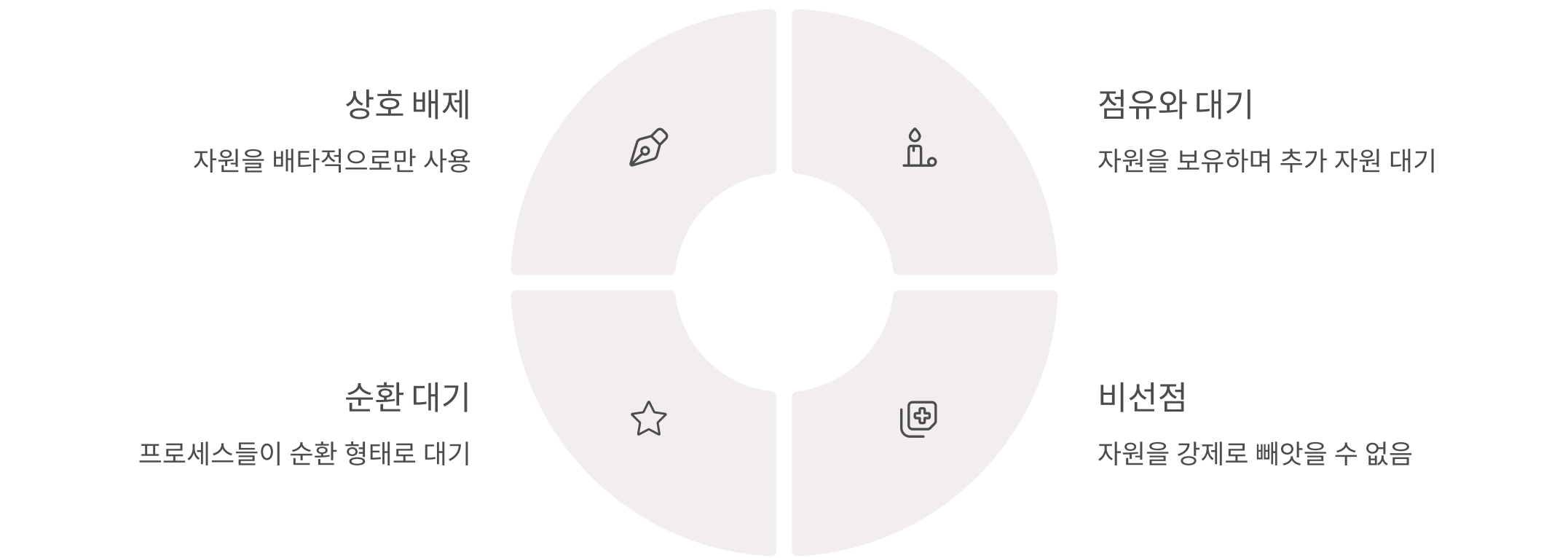
해결 조건

- 1. 상호 배제: 한 번에 하나의 프로세스만 실행
- 2. 진행: 무한 대기 방지
- 3. 한정 대기: 유한 시간 내 접근 보장

동기화 도구

무텍스 (Mutex) 이진 락으로 한 번에 하나의 스레드만 접근 허용. 소유권 개념이 있어 락을 건 스레드만 해제 가능합니다.	세마포어 (Semaphore) 카운터 기반 동기화 도구. 이진/계수 세마포어로 구분되며, 지정된 개수만큼의 스레드 접근을 허용합니다.	모니터 (Monitor) 공유 데이터와 프로시저, 동기화를 하나로 묶은 고수준 구조체. 조건 변수를 통해 wait/signal 연산 제공합니다.
--	--	--

교착상태 (Deadlock)



위 네 조건이 모두 만족될 때 교착상태가 발생합니다.

예방 (Prevention)
4가지 조건 중 하나를 제거하여 교착상태 원천 차단

회피 (Avoidance)
은행원 알고리즘으로 안전 상태 유지

탐지 및 회복
교착상태 발견 후 프로세스 종료로 해결

5. 메모리 관리

메모리 계층 구조



가상 메모리의 장점

- 프로세스 크기 제한 완화
물리 메모리보다 큰 프로그램 실행 가능
- 메모리 보호 강화
프로세스 간 주소 공간 완전 분리
- 메모리 공유 효율성
코드 영역 공유로 메모리 절약
- 프로그램 로딩 최적화
필요한 부분만 선택적 로딩

페이징 vs 세그멘테이션

특성	페이징	세그멘테이션
분할 단위	고정 크기 페이지	가변 크기 세그먼트
단편화	내부 단편화	외부 단편화
주요 장점	외부 단편화 해결	논리적 구분 명확
구현 복잡도	상대적으로 간단	복잡한 관리 필요

페이지 교체 알고리즘

1

FIFO
가장 오래된 페이지 교체. 벨라디 이상현상 발생 가능

2

LRU
가장 오래 사용되지 않은 페이지 교체. 지역성 원리 활용

3

최적 (Optimal)
미래에 가장 늦게 사용될 페이지 교체. 이론적 최적

4

Clock
FIFO + 참조 비트. LRU 근사하며 실용적 구현

스래싱 (Thrashing)

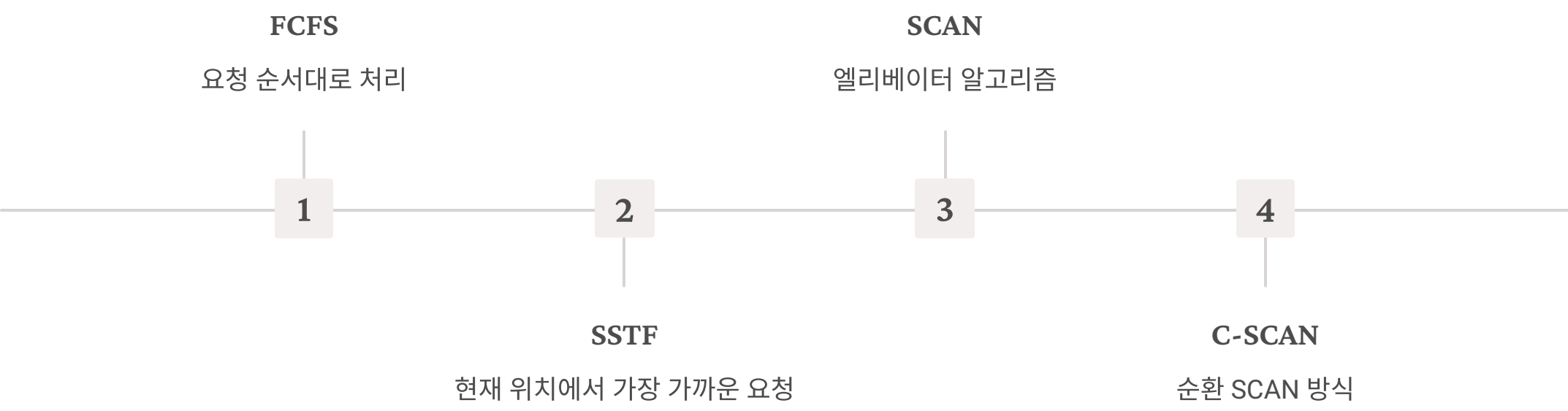
페이지 부재가 빈번하게 발생하여 CPU 성능이 급격히 저하되는 현상입니다. 충분한 프레임 할당과 작업 집합 모델 적용으로 해결할 수 있습니다.

6. 파일 시스템 & 7. 최신 기술

파일 할당 방법

연속 할당 파일을 연속된 디스크 블록에 저장. 빠른 접근이지만 외부 단편화 발생	연결 할당 링크드 리스트 구조로 저장. 단편화 해결 하지만 순차 접근만 가능	색인 할당 인덱스 블록으로 위치 관리. 직접/간접 접근 모두 지원
--	--	--

디스크 스케줄링




가상화 vs 컨테이너

특성	컨테이너	가상머신
격리 수준	프로세스 수준	하드웨어 수준
운영체제	호스트 OS 공유	독립적 게스트 OS
시작 시간	초 단위	분 단위
자원 사용량	적음	많음
적용 분야	마이크로서비스	완전한 격리 필요


Kubernetes 핵심 구성 요소



Pod
가장 작은 배포 단위로 하나 이상의 컨테이너를 묶은 그룹



Service
Pod 집합에 대한 네트워크 엔드포인트와 로드 밸런싱 제공



Deployment
애플리케이션의 배포, 업데이트, 롤백을 관리하는 컨트롤러



ConfigMap/Secret
애플리케이션 설정 정보와 민감한 데이터를 안전하게 관리

8. 면접 핵심 질문

1	<p>프로세스와 스레드의 차이점은?</p> <p>프로세스는 독립적인 메모리 공간을 가진 실행 단위이며, 스레드는 프로세스 내에서 메모리를 공유하는 실행 단위입니다. 스레드는 생성/전환 비용이 낮고 통신이 쉽지만, 동기화 문제가 발생할 수 있습니다.</p>
2	<p>교착 상태란 무엇이며, 해결 방법은?</p> <p>두 개 이상의 프로세스가 서로의 자원을 무한정 기다리는 상황입니다. 상호배제, 점유와대기, 비선점, 순환대기 4가지 조건이 모두 만족될 때 발생합니다. 해결방법으로는 예방, 회피(은행원 알고리즘), 탐지 및 회복이 있습니다.</p>
3	<p>가상 메모리의 장점과 페이징 기법은?</p> <p>가상 메모리는 물리 메모리보다 큰 주소 공간 제공, 프로세스 간 보호, 메모리 공유 효율성 등의 장점이 있습니다. 페이징은 고정 크기 페이지로 메모리를 관리하여 외부 단편화를 해결하는 기법입니다.</p>
4	<p>뮤텍스와 세마포어의 차이점은?</p> <p>뮤텍스는 한 번에 하나의 스레드만 접근 가능한 이진 락이고, 세마포어는 지정된 개수만큼의 스레드가 접근 가능한 카운터 기반 동기화 도구입니다. 뮤텍스는 소유권 개념이 있어 락을 건 스레드만 해제할 수 있습니다.</p>
5	<p>CPU 스케줄링에서 라운드 로빈의 특징은?</p> <p>각 프로세스에게 동일한 시간 할당량을 주고 순환하여 실행하는 선점형 스케줄링입니다. 공정성을 보장하고 응답시간이 좋지만, 문맥 교환 오버헤드가 발생하며 타임 슬라이스 크기 설정이 중요합니다.</p>
6	<p>페이지 교체에서 LRU 알고리즘이란?</p> <p>Least Recently Used로 가장 오랫동안 사용되지 않은 페이지를 교체하는 알고리즘입니다. 지역성 원리를 활용하여 실제로 좋은 성능을 보이지만, 구현이 복잡하고 시간/공간 오버헤드가 있습니다.</p>
7	<p>컨텍스트 스위칭이란 무엇인가?</p> <p>CPU 제어권이 한 프로세스에서 다른 프로세스로 넘어가는 과정입니다. 현재 프로세스 상태를 PCB에 저장하고, 새로운 프로세스 상태를 복구합니다. 타이머 인터럽트, I/O 블록, 높은 우선순위 프로세스 도착 시 발생합니다.</p>
8	<p>시스템 콜이 필요한 이유는?</p> <p>보안과 안정성을 위해 사용자 모드에서는 하드웨어에 직접 접근할 수 없기 때문입니다. 시스템 콜을 통해 커널 모드로 전환하여 OS가 제공하는 서비스(파일 I/O, 프로세스 생성 등)를 안전하게 이용할 수 있습니다.</p>

면접 팁

각 개념의 **정의, 동작 원리, 장단점, 실제 사용 예시**를 체계적으로 정리하여 답변하세요. 이론과 실무를 연결하여 설명할 수 있다면 더욱 좋은 평가를 받을 수 있습니다.