3단원. CPU 스케줄링

> 대상: 전공자 · 정보처리기사 취득자 | 목적: 인프라 직무 기술 면접 대비 중요도: ★★★ 최빈출 · ★★ 자주 출제 · ★ 기본 개념

Q1. CPU 스케줄링이 필요한 이유와 평가 기준을 설명하세요. ★★★

답안 CPU는 한 번에 하나의 작업만 실행할 수 있는데 준비 상태의 프로세스는 여러 개이므로, 어떤 프로세스에 CPU를 줄지 결정해야 합니다. 목표는 CPU 이용률과 처리량(throughput)의 극대화, 그리고 총처리 시간(turnaround time), 대기 시간(waiting time), 응답 시간(response time)의 최소화입니다. 시스템 성격에 따라 우선하는 지표가 다릅니다. 배치 시스템은 처리량, 대화형 시스템은 응답 시간, 실시간 시스템은 마감 시간(deadline) 준수가 핵심입니다.

관련 개념 처리량, 총처리/대기/응답 시간, 공정성, 기아(starvation)

Q2. 선점형과 비선점형 스케줄링의 차이는 무엇인가요? ★★★

답안 비선점형은 프로세스가 CPU를 자발적으로 놓을 때(종료, I/O 대기)까지 빼앗지 않는 방식으로, 구현이 단순하고 컨텍스트 스위칭이 적지만 긴 작업이 CPU를 독점하면 응답성이 나빠집니다(FCFS, SJF, HRN). **선점형**은 타이머 인터럽트나 더 높은 우선순위 프로세스의 도착 시 CPU를 강제로 회수하는 방식으로, 응답성이 좋아 현대 범용 OS는 모두 선점형입니다(RR, SRTF, 선점 우선순위, MLFQ). 대신 컨텍스트 스위칭 오버헤드와 공유 데이터 동기화 문제가 따릅니다.

관련 개념 타이머 인터럽트, 디스패처, 컨텍스트 스위칭 오버헤드

Q3. FCFS 스케줄링의 동작과 문제점을 설명하세요. ★★

답안 먼저 도착한 순서대로 CPU를 주는 비선점 방식입니다. 구현이 큐 하나로 단순하고 기아가 없지만, **호위 효과(convoy effect)**가 문제입니다. CPU 버스트가 긴 프로세스 뒤에 짧은 프로세스들이 줄줄이 대기하여 평균 대기 시간이 크게 늘어나고, I/O 장치들도 놀게 됩니다. 대화형 시스템에는 부적합합니다.

관련 개념 호위 효과, 평균 대기 시간 계산(간트 차트), 비선점

Q4. SJF와 SRTF를 설명하고, 실제로 쓰기 어려운 이유를 말해보세요. ★★★

답안 SJF는 다음 CPU 버스트가 가장 짧은 프로세스를 먼저 실행하는 비선점 방식으로, 평균 대기 시간이 이론적으로 최소입니다. **SRTF**는 그 선점형 버전으로, 남은 실행 시간이 더 짧은 프로세스가 도착하면 선점합니다. 실제로 쓰기 어려운 이유는 ① 다음 CPU 버스트 길이를 미리 알 수 없어 과거 버스트의 지수 평균으로 예측할 수밖에 없고, ② 긴 작업이 계속 밀려 **기아**가 발생할 수 있기 때문입니다. 기아는 에이징(aging)으로 완화합니다.

관련 개념 지수 평균 예측, 기아, 에이징, 최적성 증명

Q5. 라운드 로빈(RR) 스케줄링과 타임 퀀텀 크기의 트레이드오프를 설명하세요. ★★★

답안 준비 큐를 원형으로 돌며 각 프로세스에 동일한 타임 퀀텀만큼 CPU를 주고, 퀀텀이 끝나면 큐 뒤로 보내는 선점형 방식입니다. 모든 프로세스가 공평하게 CPU를 받아 응답 시간이 보장되므로 시분할 시스템의 기본입니다. 퀀텀이 **너무 크면** FCFS와 다를 바 없어 응답성이 나빠지고, **너무 작으면** 컨텍스트 스위칭 오버헤드 비중이 커져 처리량이 떨어집니다. 일반적으로 CPU 버스트의 80% 정도가 퀀텀 안에 끝나도록 수 ms~수십 ms로 설정합니다.

관련 개념 타임 퀀텀, 응답 시간 상한, 컨텍스트 스위칭 비용

Q6. 우선순위 스케줄링의 기아 문제와 해결책은? ★★★

답안 우선순위가 높은 프로세스부터 실행하면, 높은 우선순위 프로세스가 계속 유입될 경우 낮은 우선순위 프로세스가 무한정 대기하는 **기아**가 발생합니다. 해결책은 **에이징**으로, 대기 시간이 길어질수록 우선순위를 점진적으로 올려 언젠가는 실행되도록 보장하는 것입니다. HRN(Highest Response Ratio Next)도 (대기시간+서비스시간)/서비스시간으로 우선순위를 계산해 기아를 완화하는 기법입니다.

관련 개념 기아, 에이징, HRN 공식, 정적/동적 우선순위

Q7. 다단계 큐와 다단계 피드백 큐(MLFQ)의 차이를 설명하세요. ★★★

답안 다단계 큐는 프로세스를 성격별(시스템, 대화형, 배치 등)로 여러 큐에 고정 배정하고 큐마다 다른 스케줄링을 적용합니다. 큐 간 이동이 없어 유연성이 떨어집니다. **다단계 피드백 큐**는 큐 간 이동을 허용합니다. 새 프로세스는 최상위 큐에서 시작하고, 퀀텀을 다 쓰면(CPU 바운드로 판단) 아래 큐로 강등되며, I/O 바운드 프로세스는 퀀텀을 다 쓰기 전에 반납하므로 상위 큐에 남아 우선 처리됩니다. 과거 실행 행태로 프로세스 성격을 학습하는 셈이며, 기아 방지를 위해 주기적으로 전체를 최상위 큐로 올리는 부스팅을 씁니다.

관련 개념 큐 강등/부스팅, I/O 바운드 우대, 기아 방지

Q8. 리눅스 CFS(Completely Fair Scheduler)의 동작 원리를 아는 대로 설명하세요. ★★

답안 CFS는 각 프로세스의 **가상 실행 시간(vruntime)**을 추적해, vruntime이 가장 작은(=CPU를 가장 적게 받은) 프로세스를 다음에 실행함으로써 공정성을 추구합니다. 프로세스들은 vruntime을 키로 하는 **레드-블랙 트리**에 정렬되어 최소값 선택이 O(log n)입니다. nice 값은 vruntime 증가 속도의 가중치로 반영되어, 우선순위가 높으면 vruntime이 천천히 증가해 CPU를 더 받습니다. 고정 타임 슬라이스 대신 목표 지연(targeted latency)을 실행 가능 프로세스 수로 나눠 동적으로 배분합니다. 참고로 리눅스 6.6부터는 CFS가 EEVDF 스케줄러로 대체되었습니다.

관련 개념 vruntime, 레드-블랙 트리, nice 가중치, EEVDF, SCHED_FIFO/RR

Q9. 스케줄러의 종류(장기·중기·단기)를 구분해 설명하세요. ★

답안 **장기 스케줄러(작업 스케줄러)**는 어떤 작업을 메모리에 올려 준비 큐에 넣을지 결정하여 멀티프로그래밍 정도를 제어합니다. **단기 스케줄러(CPU 스케줄러)**는 준비 큐에서 다음 실행할 프로세스를 골라 디스패치하며 ms 단위로 매우 자주 실행됩니다. **중기 스케줄러**는 메모리가 부족할 때 프로세스를 통째로 디스크로 내보내고(스왑 아웃) 나중에 다시 들여오는(스왑 인) 역할을 합니다. 현대 시분할 OS에서는 장기 스케줄러가 사실상 없고 페이징 기반 가상 메모리가 중기 스케줄러 역할을 흡수했습니다.

관련 개념 멀티프로그래밍 정도, 스와핑, 디스패처

Q10. 멀티코어 환경의 스케줄링 이슈(친화성, 부하 분산)를 설명하세요. ★★

답안 멀티코어에서는 코어마다 캐시가 있어, 프로세스를 다른 코어로 옮기면 캐시에 쌓인 데이터가 무효가 되어 성능이 떨어집니다. 그래서 스케줄러는 프로세스를 가급적 같은 코어에서 실행하는 **캐시 친화성(affinity)**을 유지하려 합니다. 반면 특정 코어에만 작업이 몰리면 안 되므로 **부하 분산(load balancing)**도 필요해, 두 목표가 상충합니다. 리눅스는 코어별 런큐를 두고 주기적 밸런싱과 유휴 코어의 작업 훔치기로 절충하며, taskset/cgroup cpuset으로 특정 코어에 고정(pinning)하는 운영 기법도 있습니다(NUMA 환경에서 특히 중요).

관련 개념 코어별 런큐, 캐시 친화성, taskset/CPU pinning, NUMA, 작업 훔치기

Q11. 우선순위 역전(Priority Inversion)이란 무엇이고 어떻게 해결하나요? ★★

답안 낮은 우선순위 프로세스가 잡은 락을 높은 우선순위 프로세스가 기다리는 사이, 중간 우선순위 프로세스가 낮은 프로세스를 선점해 실행되면서 결과적으로 높은 우선순위 프로세스가 중간보다 늦어지는 현상입니다. 1997년 화성 탐사선 패스파인더 재부팅 사건의 원인으로 유명합니다. 해결책은 **우선순위 상속**(락을 쥔 프로세스에게 대기자의 높은 우선순위를 일시적으로 물려줌)과 **우선순위 상한 프로토콜**(락 획득 시 미리 정해진 상한 우선순위로 승격)입니다.

관련 개념 우선순위 상속, 우선순위 상한, 패스파인더 사례, 실시간 시스템

Q12. 스케줄링 계산 문제: FCFS와 SJF의 평균 대기 시간을 비교해보세요. ★★

답안 예: P1(버스트 24), P2(3), P3(3)이 동시에 도착. **FCFS(P1→P2→P3)**: 대기 시간 0, 24, 27 → 평균 17. **SJF(P2→P3→P1)**: 대기 시간 0, 3, 6 → 평균 3. 짧은 작업을 먼저 처리하면 평균 대기 시간이 크게 줄어드는 것을 보여주는 전형적 예시입니다. 면접에서는 간트 차트를 그려 도착 시간·버스트 시간으로 대기/반환 시간을 계산하는 문제가 나오므로, 대기 시간 = 시작 시각 − 도착 시각(선점형은 실행 구간 합산 고려), 반환 시간 = 종료 시각 − 도착 시각 공식을 정확히 쓰는 연습이 필요합니다.

관련 개념 간트 차트, 대기 시간/반환 시간 공식, 호위 효과 수치 예시
