**★ 스케줄링을 위한 Queue 종류**

|  |  |
| --- | --- |
| Job Queue | 현재 시스템 내에 있는 모든 프로세스의 집합 |
| Ready Queue | 현재 메모리 내에 있으면서 CPU를 할당 받기위해 기다리는 프로세스의 집합 |
| Device Queue | Device I/O 작업을 대기하고 있는 프로세스의 집합 |

**장기 스케줄러(Long-term Scheduler, Job Scheduler)**

- 메모리와 디스크 사이의 스케줄링

- 임시 저장된 프로세스[[1]](#footnote-1) 중 어떤 프로세스에게 메모리를 할당하여 Ready Queue로 보낼 지 결정

- 멀티 프로그래밍 제어

\* 프로세스의 상태 : New → Ready

**단기 스케줄러(Short-term scheduler, CPU Scheduler)**

- CPU와 메모리 사이의 스케줄링

- Ready Queue에 존재하는 프로세스 중 어떤 프로세스를 CPU를 할당, 실행할지 결정한다

\* 프로세스의 상태 : Ready → Running → Waiting → Ready → ∙∙∙

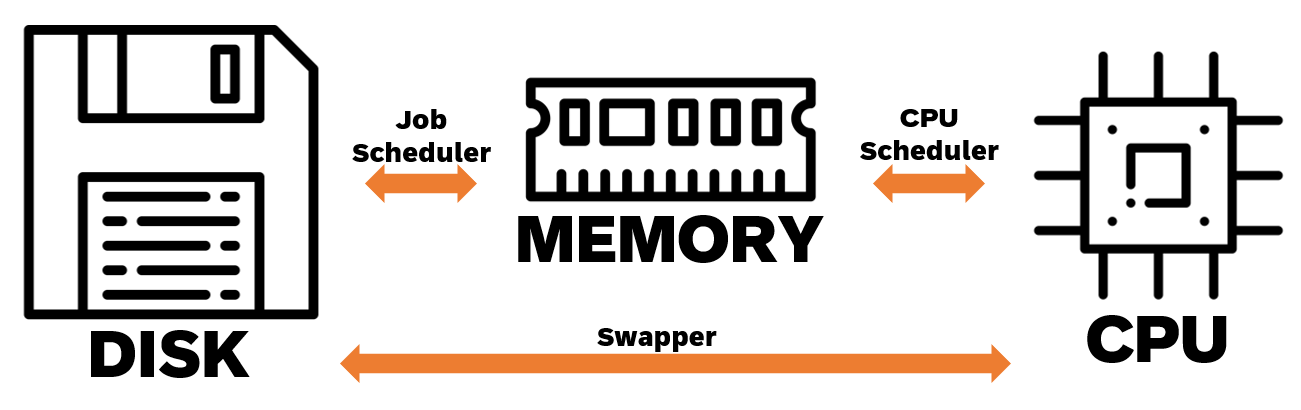
**중기 스케줄러[[2]](#footnote-2)(Medium-term scheduler, Swapper)**

- 메모리에 너무 많은 프로그램이 동시에 올라가는 것을 조절

- 여유 공간 마련을 위해 프로세스를 통째로 메모리에서 디스크로 내려씀[[3]](#footnote-3)

- 프로세스에 할당된 메모리 회수 및 멀티프로그래밍 제어

\* 프로세스의 상태 Ready→ Suspended[[4]](#footnote-4)



**★ CPU 스케줄링의 종류 : 스케줄링 대상은 Ready Queue에 있는 프로세스**

|  |  |
| --- | --- |
| 선점형(Preemptive) | 비선점형(Non-Preemptive) |
| SRTF | FCFS  SJF  우선순위 |

**FCFS(First-Come-First-Served)**

- 비선점형 스케줄링[[5]](#footnote-5)

- 먼저 실행된 프로세스를 먼저 처리한다

**\* Convoy Effect :** 소요시간이 긴 프로세스가 먼저 실행되면 효율성이 낮아진다

**SJF(Shortest-Job-First)**

- 비선점형 스케줄링

- CPU 실행시간이 짧은 프로세스를 먼저 실행함

**\* Starvation :** CPU 실행시간이 긴 프로세스는 영원히 대기하는 문제.

**SRTF(Shortest-Remaining-Time-First)**

- 선점형 스케줄링[[6]](#footnote-6)

- 새로운 프로세스가 도착할 때마다 다시 스케줄링

**\* Starvation**

**\*** 새로운 프로세스가 도착할 때마다 새롭게 스케줄링하므로 CPU 실행 시간을 측정하기 어려움

**우선순위 스케줄링(Priority Scheduling)**

- 선점형 스케줄링 : 우선순위[[7]](#footnote-7)가 가장 높은 프로세스에게 CPU를 할당

- 비선점형 스케줄링 : 더 높은 우선순위의 프로세스가 도착하면 Ready Queue의 Head에 삽입

**\* Starvation**

**\* Indefinite Blocking :** 실행 준비는 되었으나, CPU를 사용 못하는 프로세스를 CPU가 무기한 대기하는 상태

→ **해결 방법은 Aging!** 프로세스의 대기 시간에 따라 우선순위를 높여줌

**라운드로빈(Round Robin)**

- 각 프로세스에 동일한 CPU 할당 시간(Time Quantum)을 부여

- 할당 시간이 지나면 다음 프로세스가 CPU를 선점하며 실행 중이던 프로세스는 Ready Queue에 삽입

- CPU 실행 시간이 랜덤한 프로세스들이 섞여있을 경우에 효율적

- 단, TQ가 너무 커지면 FCFS와 같아지고 너무 작아지면 Context Switching으로 인한 오버헤드가 발생할 수 있다

\* Round Robin 방식이 가능한 이유는 Context Switching이 가능하기 때문이다

(예시) n개의 프로세스가 Ready Queue에 있고 할당시간이 Q인 경우

→ 각 프로세스는 Q단위로 CPU 시간의 1/n을 얻는다. 어떤 경우에도 (n-1)Q 단위 이상 기다리지 않는다

1. 한정된 메모리에 많은 프로세스가 한꺼번에 올라올 경우 대용량 메모리(≒디스크)에 임시로 저장된다 [↑](#footnote-ref-1)
2. 중기 스케줄러 내용은 WooVictory/Ready-For-Tech-Interview/Operating System/스케줄러의 종류.md 참고 [↑](#footnote-ref-2)
3. 메모리 ↔ 디스크 사이에서 swapping [↑](#footnote-ref-3)
4. Suspended 상태란 외부적인 이유로 프로세스의 수행이 정지되어 메모리에서 내려간 상태를 의미함. 프로세스가 전부 디스크로 swap 되었으며, 스스로 Ready 상태로 돌아갈 수 없다.

   (cf) Block 상태 : 다른 I/O 작업을 기다리는 상태로 스스로 Ready 상태로 돌아갈 수 있다. [↑](#footnote-ref-4)
5. 현재 실행하는 프로세스가 끝날 때까지 CPU 할당을 해지하지 않음 [↑](#footnote-ref-5)
6. 현재 실행하는 프로세스보다 CPU 사용시간이 짧거나(SRTF), 우선순위가 높은(PS) 프로세스가 도착하면 실행하던 프로세스의 CPU 할당을 해지하고 새 프로세스에 CPU를 할당 [↑](#footnote-ref-6)
7. 정수로 표현하며 숫자가 작을수록 우선ㅅ누위가 높다 [↑](#footnote-ref-7)