|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | |  | | **[ C언어 main함수에서 반환값이 0인 이유 ]** | | **프로세스를 실행하면 부모프로세스에서 커널의 fork()함수를 시스템콜하여**  **자식프로세스로 부모프로세스와 동일한 프로세스를 생성한다.**  **동일한 프로세스에서 exec()함수를 시스템콜하여 원하는 코드로 덮어씌우게된다**  **이때 자식프로세스가 반환하게되는 PID(Process ID)값은 0이다**  **부모프로세스에 0을반환하는건 정상 종료되었다는것을 알려주게된다** | |  | | **[ 선점형 ( Preemptive 프리앰티브 ) ]** | | **Cpu스케줄링 에서 여러프로세스가 실행중에 하나의 프로세스가 독점해서 사용하는것이 아니라**  **Timer의 시간을 두어서 정해진시간이 되면 cpu의 사용권을 강제로 빼앗아온다** | |  | | **[ 비선점형 ( non-Preemptive 넌프리앰티브 ) ]** | | **Cpu스케줄링 에서 여러프로세스가 실행중에 하나의 프로세스가 cpu를 독점하여 프로세스를 완료하고**  **그다음 프로세스로 cpu의 사용권을 스스로 넘긴다** | |  | | **[ 호위효과 ( Convoy Effect 콘보이 이펙트) ]** | | **1차세계대전 때 전쟁물자를 호송을 하는대 물자를 보호하기위해 앞뒤로 탱크가 호위하였던 것에서**  **유래되어 Cpu스케줄링에 있어서 FCFS알고리즘의 문제점으로 먼저들어온 프로세스의 실행시간이**  **아주 길게되면 뒤에들어오는 짧은 프로세스들이 전부 기다리게되어 전체적인 cpu성능을 저해하는 상황이다** | |  | | **[ FCFS ( First-Come First-Served 퍼스트컴 퍼스트서브드) ]** | | **프로세스들이 도착한 순서에 따라 실행한다. 그러나 먼저 온 프로세스가 오랜 시간을 요구하는 경우**  **뒤에 온 프로세스들은 대기 시간이 길어질 수 있는 단점이 있다** | |  | | **[ SJF ( Shortest-Job-First 숏테스트잡퍼스트 ) ]** | | **비선점형 방식으로**  **먼저 도착한 프로세스를 cpu버스트하고 버스트 타임안에 도착한 프로세스들 중에**  **cpu버스트타임이 짧은 순서대로 프로세스를 실행한다** | |  | |  | | **[ 디스패처 ( Dispatcher ) ]** | | **커널에 디스패처 함수가있다 cpu스케줄링으로 결정이된 후 문맥( Context )을저장하고**  **다음프로세스에게 cpu를 할당하는 역활을 해준다** | |  | | **[ 램영역 ( Ram ) ]** | | **RAM----------------------------------mode bit 1**  **| 커널영역 | 텍스트영역 | 스택영역 | 힙영역 |**  **-----------------------------------------------** | |  | | **[ 커널영역 ( Kernel ) ]** | | **커널영역----------------------------------------------------------mode bit 0**  **| 텍스트영역 | 데이터영역 | 스택영역 | 힙영역 | 프로세스 제어 블록 |**  **--------------------------------------------------------------------------** | |  | | **[ SRTF ( Shortest-Remaining-Time-First 숏테스트 리마이닝타임 퍼스트 ) ]** | | **SJF의 선점형 방식 으로**  **짧은 프로세스를 실행중에 새로운 cpu버스트타임이 더짧은 프로세스가 도착하면**  **더짧은 프로세스를 디스패처 한다** | |  | | **[ 기아 ( Starvation 스타베이션 ) ]** | | **SRTF의 방식에서 cpu버스트타임이 짧은 프로세스들이 계속해서 앞쪽에 끼어들어서**  **cpu버스트타임이 긴 프로세스는 실행되지 못하는 현상** | |  | | **[ 우선순위 스케줄링 ( Priority Scheduling 프라이올리 스케줄링 ) ]** | | **우선순위가 제일높은 프로세스에게 cpu를 준다** | |  | | **[ 에이징 ( Aging ) ]** | | **우선순위가 낮은 프로세스라도 기다림에 따라서 우선순위를 높여준다** | |  | |  | | **[ RR ( Round Robin 라운드로빈) ]** | |  | |  | | **[ 타임슬라이스 / 시간할당량 ( Time Quantum 타임 퀀텀) ]** | | **cpu가 하나의 프로세스를 몇초동안 실행하고 다음 프로세스를 디스패치할껀지의 시간이다** | |  | | **[ 멀티레벨큐 MLQ ( MultiLevel Queue ) ]** | | **단일코어를 전재로한다 우선순위큐로 프로세스를 줄을세운다**  **준비큐 ( Ready queue )는 여러 큐가 이있다**  **1) Foreground**  **I/O인터럽트,Trap인터럽트가 많은 프로세스가 있는 큐**  **2) Background**  **Cpu를 오랫동안 사용하는 프로세스가 있는 큐** | |  | | **[ 멀티레벨 피드백큐 MLFQ ( MultiLevel Feedback Queue ) ]** | | **단일코어를 전재로한다**  **1)-먼저 들어온 프로세스는 가장 높은 우선순위 큐에 들어가게 된다**  **이 큐에서는 라운드로빈 방식으로 타임 슬라이스 시간만큼 실행한다**  **만약 프로세스가 해당 타임 슬라이스 시간 안에 끝나지 않았다면**  **2)-다음으로 낮은 우선순위 큐로 이동하여 대기한다**  **그리고 다음 우선순위 큐에서는 더 긴 타임 슬라이스 시간을 할당하여 실행한다**  **만약 프로세스가 여전히 실행이 완료되지 않았다면**  **3)-이번에는 가장 낮은 우선순위 큐에 들어가서 FCFS 방식으로 프로세스를 실행한다** | |  | | **[ 멀티 프로세서 스케줄링 ( Multiple Processor Scheduling ) ]** | | **코어가 여러개인 경우** | |  | | **[ 리얼타임스케줄링 ( Real Time Scheduling ) / 리얼타임잡 ( Hard Real Time Task ) ]** | | **주어진 시간에 데드라인 안에 반드시 끝내야하는 프로세스 스케줄링 방식** | |  | | **[ 스레드 스케줄링 ( Thread Scheduling ) ]** | | **사용자레벨(유저스레드)는 사용자프로그램 안에서 실행흐름인 스레드가 Cpu에 디스패치 할지 결정한다**  **커널스레드는 운영체제가 스레드의 존재를 알고있고 커널의 cpu스케쥴링 방식으로 cpu를 디스패치 한다** | |  | | **[ 프로세스 동기화 ( Process Synchronization 프로세스 싱크로나이제이션 ) 문제 ]** | | **공유데이터( Shared Data )에 동시접근( Concurrent Access ) 하면**  **데이터처리에 시간적 차이가 생겨서 원치않는 결과가 나온다**  **커널모드를 실행중일때는 타임슬라이스가 다되어도 커널모드에서 유저모드로 바뀔때까지는 실행한다**  **커널모드에서 변경하는 데이터가 동일한게 아니라면 허용하고 동일하다면 락을 걸어준다** | |  | | **[ 임계영역 ( Critacal Section 크리티컬 영역 ) ]** | | **공유데이터를 말하는것이 아니라 공유데이터를 접근한 코드를 말한다** | |  | | **[]** | | **-** | |  | | **[]** | | **-** | |  | | **[]** | | **-** | | | |