**2022학년도 1학기 중간과제물(온라인제출용)**

**교과목명 : 운영체제**

**학 번 : 202034-153746**

**성 명 : 이동열**

**연 락 처 : 010-5264-5565**

**과제유형(공통형/지정형) : 공통형**

1.

(1)

운영체제 이름 : macOS

운영체제 버전 : Big Sur 11.6

(2)

컴퓨터 시스템 이름 : iPhone8

운영체제 이름 : IOS

운영체제 버전 : 15.3.1

2.

(1)

SJF는 Shortest Job First Scheduling의 약자로 최단 작업 우선이라는 뜻을 가지고 있는 알고리즘이다. 이 알고리즘은 준비 큐에서 대기하고 있는 프로세스들을 확인하여 CPU 점유시간을 기준으로 가장 짧은 순서대로 디스패치 한다. 이 알고리즘은 프로세스가 자원을 할당 받았을 때 해당 프로세스가 스스로 자원을 반납하기 전까지 자원을 점유하고 있는 비선점 스케줄링 알고리즘이다. 이 알고리즘은 배치환경에서 구현하기 쉬우며 점유시간을 기준으로 하기 때문에 CPU 점유시간이 미리 주어지는게 특징이다. 이 알고리즘의 단점은 특정 프로세스가 CPU 점유시간이 길 경우 먼저 준비 큐에 도착하더라도 이후에 도착한 큐에게 우선순위를 계속 뺏기게 될 수도 있다. 또한 CPU 점유시간에 대한 정보를 얻는 것이 힘들기 때문에 이런 문제들을 종합하면 장기보다는 단기 스케줄링에 적합하고 대화형 시스템에는 잘 맞지 않다는 것을 확인 할 수 있다.

SJF 알고리즘으로 반환시간과 대기시간을 구하고 싶을 경우 아래와 같은 과정으로 구할 수 있다. 먼저 아래처럼 4개의 프로세스에 대해 도착시간과 점유시간 정보를 알고 있다고 가정한다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 프로세스 | A | B | C | D |
| 도착시간 | 0 | 5 | 2 | 1 |
| CPU 점유시간 | 4 | 1 | 2 | 6 |

대기시간은 준비 큐에 도착해서 디스패치되기 전까지의 시간이다. A는 도착하자마자 디스패치 되었으므로 0이고 점유시간은 4이므로 반환시간은 4가 된다. A가 반환되기 전에 C,D가 준비 큐에 도착하게 되는데 D보다 C가 점유시간이 더 짧기 때문에 C가 먼저 디스패치 된다. 이때 C의 대기시간은 C이전 총 CPU 점유 시간에서 도착시간을 빼면 되므로 4 – 2 해서 2가 되고 반환시간은 C포함 총 CPU 점유 시간에서 도착시간을 뺀 6-2 = 4가 된다. 이렇게 C까지 종료되면 B가 준비 큐에 도착해 있고 B와 D중 점유시간이 짧은 것은 B이기 때문에 B가 먼저 디스패치 된다. 대기시간은 (4+2) – 5 = 1이고 반환시간은 (4+2+1) – 5 = 2가 된다. 마지막으로 D의 대기시간은 (4+2+1) – 1 = 6이 되고 반환시간은 (4+2+1+6) – 1 = 12가 된다.(아래 표)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 프로세스 | A | B | C | D |
| 대기시간 | 0 | 1 | 2 | 6 |
| 반환시간 | 4 | 2 | 4 | 12 |

만약 평균 대기시간과 평균 반환시간을 구하고 싶다면 각 항목을 더해서 프로세스 수로 나눠주면 된다. 그렇게 할 경우 평균 대기시간은 (0+1+2+6)/4 = 2.25가 되고 평균 반환시간은 (4+2+4+12)/4 = 5.5가 된다.

HRN은 Highest Response Ratio Next의 약자로 응답비율이 가장 높은 프로세스를 먼저 디스패치하는 비선점 스케줄링 알고리즘이다. 여기서 응답비율은 (대기시간 + 점유시간) / 점유시간으로 구할 수 있으며, 점유시간이 짧거나 대기시간이 길 수록 응답비율이 높아져 먼저 디스패치 된다. 그렇기 때문에 SJF와 달리 현재 대기 중인 프로세스보다 뒤에 도착한 프로세스의 점유시간이 짧더라도 오래 대기한 프로세스가 우선적으로 디스패치 될 수 있어 기아 상태를 해결 할 수 있게 된다. 이 알고리즘으로 대기시간, 응답비율, 반환시간을 구하려면 아래와 같은 과정을 거치면 된다. 먼저 아래처럼 4개의 프로세스가 있다고 가정한다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 프로세스 | A | B | C | D |
| 도착시간 | 0 | 5 | 2 | 1 |
| CPU 점유시간 | 4 | 1 | 2 | 6 |

A부터 확인해 보면 A의 대기시간은 0이고 응답비율은 (0 + 4) / 4 = 1이다. 어차피 대기 중인 다른 프로세스가 없기 때문에 제일 먼저 디스패치 된다. A가 반환되기 전에 C,D가 준비 큐에 도착하게 되며 A 반환 기준으로 각 대기 시간은 C가 4 – 2 = 2, D가 4 – 1 =3 이다. 각 응답비율은 C가 (3+2)/2 =2.5, D가 (3+6)/6 = 1.5가 되며 C가 응답비율이 더 크기 때문에 C부터 디스패치 된다. 이때 C의 반환 시간은 (4+2) -2 = 4가 된다. C가 실행중인 동안 B가 도착하게 되고 B와 D의 각 대기 시간은 B가 (4+2)-5 = 1, D가 (4+2) – 1 =5이며, 응답 비율은 B가 (1+1)/1 = 2, D가 (5+6)/6 = 1.8333..이 된다. 여기서 응답비율은 B가 더 크기 때문에 B부터 디스패치 되며 B의 반환시간은 (4+2+1)-5 = 2가 된다. 마지막으로 D의 대기시간은 (4+1+2) – 1 = 6이 되고 응답비율은 (6+6)/6 = 2, 반환시간은 (4+1+2+6)-1 = 12가 된다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 프로세스 | A | B | C | D |
| 대기시간 | 0 | 1 | 2 | 6 |
| 응답비율 | 1 | 2 | 2.5 | 2 |
| 반환시간 | 4 | 2 | 4 | 12 |

(2)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 도착시간 | 0 | 2 | 4 | 5 | 9 |
| 프로세스 | A | B | C | D | E |
| CPU 사이클 | 7 | 4 | 1 | 3 | 2 |

먼저 SJF로 실행 순서와 평균 대기시간을 구해보면, 일단 A가 가장 먼저 도착하므로 A가 바로 디스패치 된다. 바로 디스패치 됐으므로 대기시간은 0이고 실행 순서도 첫 번째가 된다. A는 CPU사이클이 7이므로 준비 큐에는 도착시간이 7이하인 프로세스가 도착하게 된다. 도착하는 프로세스는 B,C,D이고 이 프로세스들 중 CPU사이클이 가장 짧은 C가 실행된다. C의 대기시간은 C가 디스패치 되기 전 총 사이클에서 C의 도착시간을 제외하면 된다. 그렇게 하면 7-4가 되어 C의 대기시간은 3이 된다. C가 끝나더라도 사이클이 9가 넘지 않기 때문에 E는 도착하지 않은 상태이다. 그렇기 때문에 다음으로 디스패치 될 프로세스는 D가 된다. C에서 대기시간을 구했을 때와 같은 방법으로 대기시간을 구하면 (A사이클 + C사이클) – D도착시간 이런식으로 구할 수 있다. 따라서 D의 대기시간은 (7+1)-5 = 3이 된다. D가 끝나면 총 사이클은 11이 되기 때문에 준비 큐에는 E가 도착해 있게 된다. 그리고 E와 B 중에 더 짧은 사이클을 가지고 있는 프로세스는 E이므로 E가 먼저 디스패치 된다. E의 대기시간은 (A사이클+C사이클+D사이클)-E도착시간으로 값을 대입하면 (7+1+3)-9=2가 된다. 이제 마지막으로 B의 대기시간은 (B외 모든 사이클 합) – B도착시간으로 구할 수 있으며 값을 대입하면 (7+1+3+2)-2=11이 된다. 이 과정을 정리하면 아래 표처럼 되며, 평균 대기시간은 (모든 대기시간)/프로세스 수로 구할 수 있다. 여기에 값을 대입하면 평균 대기 시간은 (0+11+3+3+2)/5 = 3.8이 된다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 프로세스 | A | B | C | D | E |
| 실행순서 | 1 | 5 | 2 | 3 | 4 |
| 대기시간 | 0 | 11 | 3 | 3 | 2 |

SRT는 선점 방식이기 때문에 CPU를 선점하고 있는 프로세스가 있더라도 이후에 준비 큐에 도착한 프로세스의 사이클이 현재 선점하고 있는 프로세스의 남은 사이클보다 짧다면 실행 순서가 바뀌게 된다. 이런 방식으로 순서와 대기시간을 구해보면, A가 제일 먼저 도착하게 되므로 A가 바로 디스패치 된다. 이때 A의 사이클이 2가 되면 B가 도착하게 되는데 B의 사이클은 4이고 A의 남은 사이클은 5이므로 B가 디스패치 된다. 이후에 B도 사이클이 2가 되면 C가 도착하고, C의 사이클은 1, B의 남은 사이클은 2이기 때문에 C가 디스패치 된다. 그 다음 C가 종료되면 D가 준비 큐에 도착해 있는데, 이때 A,B,D의 각각 사이클은 5, 2, 3이므로 B가 다시 디스패치 되고 B가 끝나게 되면 A와 D가 남게 되어 다음으로 D가 디스패치 된다. 이 상태에서 D가 2사이클이 되면 E가 준비 큐에 도착하지만 D의 남은 사이클이 E의 사이클보다 짧기 때문에 그대로 D가 계속 실행되게 된다. 이렇게 D도 끝나면 A와 E만 남게 되는데 A의 사이클은 5, E의 사이클은 2 이므로 E가 디스패치되고 그 다음에 A가 디스패치 된다. 이 과정을 표로 작성하면 아래처럼 작성 할 수 있다.

|  |  |  |
| --- | --- | --- |
| 프로세스 | 시작 | 남은 사이클D |
| A | 0 | 5 |
| B | 2 | 2 |
| C | 4 | 0 |
| B | 5 | 0 |
| D | 7 | 0 |
| E | 10 | 0 |
| A | 12 | 0 |

위 표를 참고해 대기시간을 구해보면 2부터 12까지 대기 했으므로 10, B는 4부터 5까지 대기했으므로 1, C는 도착하자마자 디스패치돼서 교체없이 종료되었기 때문에 0, D는 5에 도착해서 7에 시작했고 교체없이 종료되었으므로 2, E는 9에 도착해서 10에 시작했고 교체 없이 종료되었으므로 1이다. 이 내용을 표로 정리하면 아래처럼 나오게 작성 할 수 있다. 또한 평균 대기시간은 총 대기시간에서 전체 프로세수 수를 나눠 구할 수 있고 여기에 아래 값들을 넣어 계산하면 (10+1+0+2+1)/5 = 2.8이 된다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 프로세스 | A | B | C | D | E |
| 대기시간 | 10 | 1 | 0 | 2 | 1 |