운영체제 프로그래밍 과제

Simulator for Memory Allocation Algorithm

문제 :

주 메모리 할당을 위한 간단한 simulator를 만들려고 한다. 메모리에 적재되어 실행되어야 할 프로세스에 대한 정보가 주어질 때, 메모리 어느 영역에 그 프로세스가 할당되는지를 결정하고자 한다. 메모리 할당을 위한 알고리즘은 크게 'first bit', 'best fit', 'worst fit'을 사용한다.

메모리에 적재되어 실행될 각 프로세스에 대한 정보는 다음과 같이 세 가지 항목으로 구성된다.

(요청시각, 실행시간, 크기)

여기서, 요청시각은 프로세스가 실행을 요청하는 시각을 나타내며, 실행시간은 그 프로세스가 메모리에 얼마나 오래 동안 적재되어 실행되어야 하는 정보이며, 크기는 프로세스의 사이즈를 말한다. 즉, 메모리가 얼마나 필요한지를 나타낸다. 프로세스의 고유번호 즉, PID는 요청순서대로 0부터 차례로 자동으로 부여된다. 가장 먼저 메모리에 적재되길 요청한 프로세스의 번호는 0이다.

예를 들어, 3개의 프로세스에 대한 정보가 아래와 같다면

(0, 10, 100)

(3, 25, 200)

(6, 12, 150)

프로세스 0은 시각 0에 메모리에 적재되길 원하며, 이 때 필요한 크기는 100이다. 일단 메모리에 적재되면 10만큼의 시간 동안 메모리에 상주하게 되고, 10 만큼의 시간이 지나면 프로세스 0은 작업을 종료하게 되어 그 메모리 영역이 반환된다. 유사하게 프로세스 1은 시각 3에 요청되었고, 메모리에 적재된 후 25만큼의 시간을 소비한 후 작업을 완료하게 되고, 그 땐 메모리 영역을 반환하게 된다.

시각 t에서 메모리의 상태는 다음과 같이 나타내자.

<주소, PID, 크기><주소,PID,크기>...

여기서, 주소는 메모리의 주소를, PID는 그 공간을 사용하고 있는 프로세스 번호(만약 그 공간이 hole이면 이 값은 -1), 크기는 공간의 크기를 나타낸다.

메모리의 크기는 1000이라고 가정할 때, 위에서 보인 입력에 대해 fist fit 알고리즘을 사용하여 메모리를 할당할 때, 시간대 별로 메모리 상태를 보이면 다음과 같다.

$t=0$: <0 0 100><100 -1 900> // P_0가 주소 0에 적재됨

$t=3$: <0 0 100><100 1 200><300 -1 700> // P_1이 주소 100에 적재됨

$t=6$: <0 0 100><100 1 200><300 2 150><450 -1, 550> // P_2가 주소 300에 적재됨

$t=10$: <0 -1 100><100 1 200><300 2 150><450 -1, 550> // P_0가 종료, 따라서 그 공간이 hole이 됨

$t=18$: <0 -1 100><100 1 200><300 -1, 700> // P_2가 종료

$t=28$: <0 -1 1000> // P_1이 종료

운영체제 프로그래밍 과제

할당 알고리즘에 따라 요청된 프로세스가 적재되는 메모리 위치가 다를 수 있다. 또한, 상황에 따라 어떤 프로세스의 요청은 그 시점에서 공간 부족으로 할당이 보류될 수 있다. 여유 공간 부족으로 공간 할당이 보류된 경우, 시스템 상황이 변화될 때 까지 운영체제는 보류된 프로세스를 대기 큐에서 기다리게 한다. 일정 시간이 지나면 어떤 프로세스가 작업을 종료하게 되고, 그러면 그 프로세스가 사용하던 공간이 반환된다. 이때 OS는 대기 큐에서 기다리고 있는 프로세스들을 순서대로 차례로 검사하여 적재 가능한 것들은 모두 메모리에 적재한다. 대기 큐에 먼저 들어온 것이 사정상 적재되지 못하고 큐에 뒤에 들어온 것이 메모리에 먼저 적재될 수도 있다는 점을 유의해야 한다.

시각 t에 종료하는 프로세스가 하나 이상인 경우, 모든 종료하는 프로세스로부터 메모리를 반환 받은 후에 대기큐에 있는 프로세스를 검사하여 메모리에 적재한다.

만약 어떤 프로세스가 적재를 요청한 시각이 t이고, 실행시간이 d라고 하자. 시각 t에 공간 부족으로 적재가 지연되어 실제로 메모리에 적재된 시각이 $t+s$라면 이 프로세스는 $t+s+d$에 작업을 종료하게 된다.

예를 들어, 어떤 시각 t에 메모리의 상태가 다음과 같다고 하자. 알고리즘은 best fit을 적용한다고 하자.

<0 0 100><100 -1 120><220 2 40><260 -1 130><390 4 50><440 5 200><640 6 160>

<800 7 40><840 8 140><980 -1 20>

이때 크기가 180인 P_9이 요청되면 이 요청은 수락할 수 없어 대기 큐에서 기다리게 된다. 그 후 크기가 220인 P_{10}이 요청되면 이 역시 대기 큐에서 기다리게 된다.

시간이 흘러 만약 P_2가 종료하게 되면 크기가 290(120+40+130)인 hole이 생기게 되고, OS는 대기큐를 차례로 검사하여 P_9을 적재하게 되어 메모리 상태가 다음과 같이 변하게 된다.

<0 0 100><100 9 180><280 -1 110><390 4 50><440 5 200><640 6 160>

<800 7 40><840 8 140><980 -1 20>

현재 상태에서도 대기 큐에는 P_{10}이 적재를 기다기고 있다.

또 다른 예를 들어보자, 어떤 시각 t에 메모리의 상태가 다음과 같다고 하자.

<0 0 100><100 -1 120><220 2 40><260 -1 130><390 4 50><440 -1 200><640 6 160>

<800 -1 40><840 8 140><980 -1 20>

이때 크기가 30인 P_9이 요청되면 적재 알고리즘에 따라 P_9이 적재되는 공간은 다르게 된다.

First fit 알고리즘을 적용하는 경우는 크기가 120인 첫 번째 hole에, best fit 알고리즘을 적용하는 경우는 크기가 40인 hole에, worst fit 알고리즘을 적용하는 경우는 크기가 200인 hole에 적재하게 된다.

입력 :

입력 파일의 이름은 allocation.inp이다. 첫째 줄에는 프로세스의 개수를 나타내는 n이 주어지고, 이어서 n줄에는 각 프로세스의 요청시각, 실행시간, 크기가 차례로 주어지며 각 값은 하나 이상의 공백으로 구분된다. 동일한 요청시각에 두 개 이상의 프로세스에 관한 입력이 주어질 수 있으며, 입력 자료에서 요청시각은 감소하지 않는다. 입력순서대로 PID는 0부터 n-1까지 차례로 자동으로 할당된다.

운영체제 프로그래밍 과제

각 프로세스는 메모리에 실제로 적재되는 시각부터 '실행시간'만큼 실행 된 후에 종료하게 되고, 사용하던 공간을 반환하게 된다. 앞에서 설명했듯이 상황에 따라 요청시각에 바로 적재되지 못하고 적재가 지연될 수 있음을 고려하여야 한다.

메모리의 크기는 1000으로 가정한다. n의 최대값은 1000이다.

출력 :

출력화일의 이름은 allocation.out이다. 세 가지 서로 다른 메모리 할당 알고리즘 'first fit', 'best fit', 'worst fit'을 각각 적용할 때, 프로세스 P_{n-1} 가 메모리 적재되는 위치 즉, P_{n-1} 가 적재된 메모리의 시작 주소를 출력하되 'first fit', 'best fit', 'worst fit' 각각에 대한 결과를 한 줄에 하나씩 차례로 출력하라.

예제 :

입력 예	입력 예에 대한 출력
8 0 13 100 1 22 120 4 12 130 6 17 140 8 15 150 9 9 160 11 12 170 21 2 25	0 970 640

제한조건: 프로그램은 allocation.{c,cpp,java}로 한다.