First-Come-First-Served Algorithm

1. 프로젝트 설계 구성
2. 프로그램의 입력 데이터의 구성

* 프로그램의 입력 데이터는 type, process\_id, priority, computing\_time의 4개의 필드로 구성된다.
* type: 0은 프로세스의 생성을 나타내며, process\_id, priority, computing\_time을 입력 받게된다.
* type: 1은 하나의 프로세스의 시간할당량이 다되어 스케줄이 되어야 함을 나타내며, process\_id, priority, computing\_time의 값은 0이 되며, 특별한 의미는 없다.
* type: -1은 입력 완료를 의미한다. 입력 완료 이후에는 Queue에 있는 일반 프로세스들이 우선순위에 따라 차례대로 스케줄링 된다.

1. 스케줄링 작동과정

* 프로세스는 한 개의 Queue를 가진다.
* 프로세스가 생성되면 생성되는 순서대로 Queue에 연결된다.
* 먼저 연결된 프로세스를 먼저 스케줄링을 하여 process\_id, priority, computing\_time, turn\_around\_time을 출력한다.
* 출력된 프로세스는 Queue에서 제거한다.

1. 생성되는 프로세스의 종류

* 생성되는 프로세스의 종류는 일반 프로세스 이며, 먼저 생성된 프로세스가 우선순위가 높아 먼저 스케줄링 된다.

1. 프로그램의 출력

* 프로그램의 출력 필드는 process\_id, priority, computimg\_time, turn\_around time의 4가지 필드를 가진다.
* Process\_id는 입력 받은 프로세스의 id를 출력한다.
* Priority는 프로세스가 종료되기 직전의 변경된 priority를 출력한다.
* Computing\_time은 입력시 지정된 computing\_time을 출력한다.
* Turn\_around time은 프로세스가 종료된 시점에서 프로세스가 입력된 시점의 상대시간을 뺀 값을 출력한다.

1. 프로그램 입력 예제

* 입력

0 1 25 80

0 2 15 40

0 3 8 30

1 0 0 0

0 4 12 10

0 5 22 30

0 6 28 50

1 0 0 0

0 7 5 20

0 8 3 40

0 9 13 60

1 0 0 0

0 10 24 45

-1

1. 프로그램 출력 예제

Process\_id priority computing\_time turn\_around\_time

1 25 80 80

2 15 40 120

3 8 30 150

4 12 10 160

5 22 30 190

6 28 50 240

7 5 20 260

8 3 45 300

9 13 60 360

10 24 45 405

1. 스케줄링 비교 분석 결과

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 프로세스 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |  |
| 도착시간 | 0 | 0 | 0 | 20 | 20 | 20 | 40 | 40 | 40 | 60 |  |
| 서비스시간 | 80 | 40 | 30 | 10 | 30 | 50 | 20 | 40 | 60 | 45 |  |
| 종료시간 | 80 | 120 | 150 | 160 | 190 | 240 | 260 | 300 | 360 | 405 | **평균** |
| 반환시간 | 80 | 120 | 150 | 140 | 170 | 220 | 220 | 260 | 320 | 345 | **203** |
| Tr/Ts | 1 | 3 | 5 | 14 | 5.67 | 4.4 | 11 | 6.5 | 5.33 | 7.67 | **6.36** |

1. 코드 분석

* 전역 변수

in\_time : 프로세스가 도착될때의 시간을 계산하기위한 변수

out\_time: 프로세스가 종료될때의 시간을 계산하기위한 변수

* 함수

void initialize\_queue(void): 프로세스를 초기화 하는 함수

→head\_queue를 생성하고 오른쪽, 왼쪽 링크를 NULL.

int insert\_queue(int, int, int): 프로세스를 생성하기위한 함수

→Queue가 NULL이면 가장 앞에 생성

→Queue가 NULL이 아니면 이전에 생성된 Queue의 맨 끝에 생성

int scheduling\_queue(int): 프로세스를 스케줄링하기위한 함수

→Queue에 프로세스가 있을경우 가장 앞의 연결된 프로세스의 computing\_time을 turn time(=20)만큼 뺀다. 그 결과 computing\_time이 0보다 작거나 같으면 프로그램의 출력 예제와 같이 출력하고, Queue에서 제거한다. Computing\_time이 0보다 크면 출력을 하지 않고 다음 스케줄링을 한다.

→ type이 -1이면 Queue에 연결된 모든 프로세스를 연결된 순서대로 출력하고 제거한다.

* 구조체

process\_pointer 구조체

→ pid, priority, computing\_time, turn\_around\_time, process\_in\_time의 변수를 가지며, pid, priority, computing\_time의 변수는 input\_file.txt에서 읽어오는 값을 저장한다.

→ turn\_around\_time은 프로세스의 반환시간을 구하기 위한 변수이다.

→ process\_in\_time은 프로세스가 생성된 시간을 구하기 위한 변수이다.

head\_pointer 구조체

→ Queue의 head와 왼쪽 오른쪽 링크를 만들기 위한 구조체이다

1. 결과화면

.

