Operating System

**[ Project-01 보고서 ]**

인공지능융합전공 2021311828 박수연

**1. 구현/설계 방법**

**1) 입력 파일의 형식**

**텍스트, 전자제품이(가) 표시된 사진

자동 생성된 설명** 입력으로 사용할 데이터는 .txt 파일에 저장되어 있으며, 파일의 첫 번째 줄은 프로세스의 수를 의미한다.

두 번째 줄부터 마지막 줄 까지는 각 프로세스 별 process id, arrival time, burst time 순서로 데이터가 나타난다. 같은 줄에 있는 데이터 값들은 띄어쓰기로 구분한다.

이 프로젝트에서 Process id는 Gantt chart를 출력할 때 포함되므로 최대한 길이를 짧게 설정하는 것이 좋고, 정수나 문자열 형태로 입력한다. Arrival time은 ready queue에 process가 도착하는 시간을 의미하며, 정수로 입력한다. Burst time은 프로그램이 실행하는데 걸리는 시간을 의미하며, 역시 정수로 입력한다.

|  |  |  |
| --- | --- | --- |
| Process id | Arrival time | Burst time |
| 최대한 짧은 길이의  숫자 또는 문자  예) a, 0, ㄱ | 양의 정수  예) 0, 14, 2 | 양의 정수  예) 0, 3, 7 |

**2) MFQ 스케줄링 방식**

MFQ(Multi-level Feedback Queue) 스케줄링 방식은 여러 개의 우선순위가 있는 ready queue를 사용하며, 우선순위가 높은 ready queue가 비면 다음으로 우선순위가 높은 ready queue에서 process scheduling이 시작된다.

이 프로젝트에서는 3개의 Ready Queue Q0, Q1, Q2를 사용하고 순서대로 높은 우선순위를 갖는다. Q0는 time quantum이 2인 RR 스케줄링 기법을 사용하며, Q1는 time quantum이 4인 RR 스케줄링 기법을 사용하고, Q2는 FCFS 스케줄링 기법을 사용한다.

또한 이 프로젝트에서는 각 프로세스의 단일 CPU burst에 대해서만 다루며, 프로세스의 I/O 입출력으로 인한 상황은 고려하지 않는다.

<MFQ Scheduling Mechanism> - dynamic priority and preemptive scheduling

① 생성된 process는 Q0에 들어간다.

② Qi에 있던 process가 time runout되면, preemption 된 process는 우선순위가 한 단계 낮은   
Qㅑ+1에 들어간다. (RR scheduling)

③ 마지막 Q2에 있는 process들은 FCFS 기법을 통해 스케줄링 된다.

**3) MFQ 스케줄링 구현 방법**

MFQ 스케줄러는 mfq.py 파일에 함수로 정의되어 있다. 함수의 이름은 MFQScheduler(f\_name)이고, 함수의 parameter는 입력할 데이터 파일의 이름이다.

① 데이터를 저장할 리스트 생성

Process ID, AT, BT, TT, WT 리스트를 생성한다.

② 입력 데이터 불러오기

입력 데이터 파일에 있는 텍스트를 한 줄 씩 불러와 변수나 리스트에 저장한다. 첫 번째 줄에 있는 프로세스 수에 대한 값은 p\_num 변수에 저장되며, 나머지 입력 데이터에 있는 값들은 column 별로 pid\_list, at\_list, bt\_list에 저장된다.

③ 필요한 변수 및 ready queue 생성하기

t : cpu time을 의미한다.

cpu\_wt : ready queue에 어떤 process도 존재하지 않아 현재 cpu에 아무 process도 처리하지   
않는 상태의 time

cpu : cpu timeline을 기록하는 리스트로 [[process\_id, run\_time],[process\_id, run\_time], ...] 과 같은  
형태로 저장

result : 각 process별 실행 종료 시간을 저장하는 딕셔너리로 process id를 key 값으로,   
process 실행 종료 시간을 value값으로 사용

not\_ready : arrival time 오기 전 즉, 처음 ready queue에 들어가기 전 까지 process id를   
저장하는 리스트

original\_bt\_list : bt\_list는 MFQ 스케줄링 과정에서 갱신되므로, 원래 bt를 저장하고 있는 리스트

pid\_at\_dict : process id를 key 값으로, arrival time을 value 값으로 가지는 딕셔너리

sorted\_pid\_at : pid\_at\_dict를 value값인 arrival time을 기준으로 오름차순 정렬한 딕셔너리

rq0 : Q0 나타내는 리스트로 .append()와 .pop(0)을 이용하여 ready queue로 사용

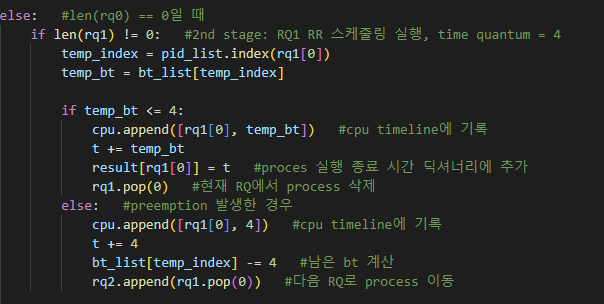
rq1 : Q1 나타내는 리스트로 .append()와 .pop(0)을 이용하여 ready queue로 사용

rq2 : Q2 나타내는 리스트로 .append()와 .pop(0)을 이용하여 ready queue로 사용

④ AT와 BT를 이용하여 MFQ Scheduling 구현

* [Stage 0] Arrival time이 짧은 시간 순서대로 저장되어 있는 sorted\_pid\_at 딕셔너리를 이용하여 cpu time이 t 일 때, arrival time이 지나 있으면 Q0에 process를 할당한다.  
  텍스트이(가) 표시된 사진

  자동 생성된 설명
* [Stage 1] cpu time이 t 일 때, 우선순위가 높은 ready queue인 rq0에 process가 있으면, time quantum이 2인 RR 스케줄링을 실행한다.  
  텍스트이(가) 표시된 사진

  자동 생성된 설명
* [Stage 2] cpu time이 t 일 때, 우선순위가 높은 rq0가 비어 있으면 그 다음으로 우선순위가 높은 rq1에 있는 process 들을 time quantum이 4인 RR 스케줄링을 한다.  
  
* [Stage 3] cpu time이 t 일 때, 우선순위가 높은 rq0, rq1이 모두 비어 있으면, 가장 우선순위가 낮은 rq2에 있는 process들이 FCFS 스케줄링된다.  
  텍스트이(가) 표시된 사진

  자동 생성된 설명
* [Stage 4] 아직 ready 상태에 오지 못한 process가 존재하지만 ready queue가 비어 있는 경우 cpu에는 아무 process가 존재하지 않는다. 이러한 상황 동안 cpu에 “-“로 표시된다.  
  텍스트, 모니터, 화면, 장치이(가) 표시된 사진

  자동 생성된 설명

⑤ Turnaround Time 계산

result에 저장되어 있는 각 process 별 실행 종료까지 걸린 시간에서 각 프로세스 별 arrival time을 빼면 각 프로세스 별 turnaround time을 계산할 수 있다. 이렇게 구한 각 프로세스 별 turnaround time은 앞에서 생성되었던 TT 리스트에 저장된다.

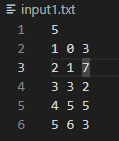
⑥ Waiting Time 계산

Waiting time = Turnaround time – Burst time이므로, TT 리스트와 BT 리스트에 저장되어 있는 각 프로세스별 값을 활용하여 waiting time을 계산하고 WT 리스트에 저장한다.

⑦ WT와 TT 평균 계산

WT 리스트와 TT 리스트를 이용하여 WT와 TT의 평균을 계산한다.

**2. 다양한 입력에 대한 실행 결과  
1) input1.txt - 기본 예시**



텍스트이(가) 표시된 사진

자동 생성된 설명

**2) input2.txt – process id 변경하기**

텍스트, 전자제품, 계산기이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

**3) input3.txt – arrival time 순서 정렬이 되어 있지 않고, arrival time이 0이 없는 경우**

**텍스트, 전자제품이(가) 표시된 사진

자동 생성된 설명**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**4) input4.txt – cpu에 어떤 process도 들어와 있지 않은 경우**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**5) input4.txt – arrival time이 같은 process가 있는 경우**

**텍스트, 전자제품이(가) 표시된 사진

자동 생성된 설명**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**3. 실행 결과 확인 방법**

mfq.py에서 MFQScheduler 함수를 불러와서 출력 결과를 확인할 수 있다. MFQScheduler 함수에 스케줄링르 하고자 하는 파일의 파일명을 입력하면, 스케줄링 결과 (Gantt chart), 각 프로세스 별 turnaround time과 waiting time 그리고 전체 프로세스의 turnaround time과 waiting time 평균을 확인할 수 있다.

이 프로젝트에서는 총 7개의 input text file을 사용하였는데, project1\_main.py에서 전체 결과를 한 번에 확인할 수 있다.

다음은 project1\_main.py 파일에서 MFQ Scheduling한 결과이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

**1) 스케줄링 결과**

이 부분에서는 gantt chart를 통해서 시간 대 별로 어떤 process가 cpu에서 실행되었는 지를 알 수 있다. 구분선 “|”를 기준으로 프로세스가 처리된다. 만일 위의 예시처럼 process id ‘1’이 cpu time 2만큼 실행된다면, id를 나타내는 칸 1칸과 “ “빈 칸 1으로 총 2칸을 통해서 “ 1”로 cpu 실행을 표현한다.

Process의 burst time이 0인 경우, Gantt chart에 나타나지 않을 수 있으므로, Gantt chart 아래에 cpu 리스트를 출력하여 cpu의 timeline을 확인할 수 있도록 한다.

**2) 각 프로세스 별 TT와 WT**

앞에서 각 프로세스 별 turnaround time과 waiting time을 저장한 리스트를 이용하여 값을 출력한다. “i번째 프로세스: TT (num) / WT (num)”의 형식으로 출력된다.

**3) [전체 프로세스의 TT 평균과 WT 평균]**

이미 계산했던 전체 프로세스의 TT 평균과 WT 평균 값을 불러온다.

“전체 프로세스의 TT 평균 : (turnaround time mean value)”,

“전체 프로세스의 WT 평균: (waiting time mean value)”의 형식으로 출력된다.